Applying SCRUM in an OSS Development Process: An Empirical Evaluation

Luigi Lavazza, Sandro Morasca, Davide Taibi, and Davide Tosi

Università degli Studi dell'Insubria
{luigi.lavazza,sandro.morasca,davide.taibi}@uninsubria.it,
davide.tosi@uninsubria.it

Abstract. Open Source Software development often resembles Agile models. In this paper, we report about our experience in using SCRUM for the development of an Open Source Software Java tool. With this work, we aim at answering the following research questions: 1) is it possible to switch successfully to the SCRUM methodology in an ongoing Open Source Software development process? 2) is it possible to apply SCRUM when the developers are geographically distributed? 3) does SCRUM help improve the quality of the product and the productivity of the process? We answer to these questions by identifying a set of measures and by comparing the data we collected before and after the introduction of SCRUM. The results seem to show that SCRUM can be introduced and used in an ongoing geographically distributed Open Source Software process and that it helps control the development process better.

Keywords: Open-source Software, OSS, agile methods, SCRUM, process improvement evaluation.

1 Introduction

The development of Open Source Software (OSS) products does not usually follow the traditional software engineering development paradigms described in textbooks. While classical development paradigms have been designed with Closed Source Software (CSS) in mind, OSS has inherent characteristics that make these paradigms hardly applicable. For example, OSS is often developed by a distributed community of developers and is freely distributed, as the source code is open and available to both developers and end-users under specific license policies.

As we now describe, an analysis of some inherent characteristics of OSS [13] on the one hand confirms that rigid development paradigms, such as the Waterfall model [14], are not applicable to OSS, while, on the other hand, it seems to suggest that agile paradigms [15] may help OSS developers improve the quality of their products. This is a result of the fact that the general principles behind the Agile Manifesto [2] are reflected in the way most OSS projects are developed and released to final users.

The short-term and non-commercial vision that characterizes many OSS projects implies that system analysis and product design are usually not pre-planned activities in OSS development. Also, many OSS projects were started to solve a user's particular problem, but they sometimes ended up deeply innovating the software field (this is the case of Linux, Perl, and the World Wide Web), even though they did not have a long-term vision, at least at their inception. The need for deep evolution of most OSS projects is usually perceived *after* they are successfully used by a large community of users. The success of an OSS product is usually unpredictable. It is directly related to the degree of attractiveness and usefulness that the project produces in the user community over time. It is thus quite hard to pre-plan the new releases of a system, except for the short term. These observations reflect the first two principles of the Agile manifesto [2]: "... satisfy the customer through early and continuous delivery of valuable software" and "Deliver working software frequently ...".

Moreover, OSS is characterized by an unstructured working environment, where the majority of OSS developers are volunteers who would not commit to hard deadlines and strict assignments. In CSS projects, team members are assigned tasks; in OSS projects, team members choose tasks. Due to this freedom, activities that are viewed as nuisances –such as project plan definition, system design evaluation, and requirements analysis– cannot be performed following traditional paradigms in the OSS community. Requirements that were not defined in advance by skilled analysts are continually discussed by the developers. Risks are monitored and managed during the project life-cycle in a natural way, as part of the "regular" development work. These OSS characteristics reflect the Agile principle "The best architectures, requirements, and designs emerge from self-organizing teams."

OSS is developed in a collaborative and distributed way [8]. OSS systems are developed in a large-scale cooperative context, where different teams, private users, a passionate core of developers, and virtual communities create the "unstructured company" (E. S. Raymond called it "Bazaar" [8]) that contributes to the project. Internet is the scaffolding and the desktop for these virtual software development organizations, and developers are coordinated by simple license policies without hierarchical supervision mechanisms as stringent as in CSS development. As a consequence, OSS developers hardly ever follow development methodologies as well defined as those followed by CSS developers. Furthermore, OSS is believed to be characterized by a faster growth [16] and more creativity than CSS [17], with the goal of satisfying and responding to user needs more quickly. This is primarily due to the unstructured and informal organization of the communities. Structure and rules may "inhibit innovative thinkers and drive them to the fringes," [17] while informality and freedom boost action and creativity. This implicitly requires the definition of architectures that are inherently modular and scalable, to guarantee the extensibility of a system and its interoperability across different hardware and software platforms. These observations reflect the seventh Agile principle "Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done."

The massive use of the network and distributed resources fosters the dissemination of project knowledge via unstructured channels like mailing lists, forums, and chat logs, thus facilitating the communication between developers, final users, and managers. This partially meets the two Agile principles: "Business people and developers must work together daily throughout the project" and "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."

Thus, the work reported in this paper moved from two main considerations: 1) the development process of OSS products often resembles Agile or XP models in which each small-grained incremental development step is performed via a cycle involving test design, and execution; 2) while in the last few years several CSS developers have dismissed rigid development paradigms and embraced Agile or XP models, few works about the application of Agile paradigms in OSS development have been released (e.g., [10]).

We applied an Agile methodology to the development of an OSS tool in the QualiPSo project [18], and we performed measurements to evaluate whether it is possible to improve process productivity and OSS product quality via an environment that is more controlled than the one completely unstructured that characterizes OSS projects. Specifically, we applied the SCRUM [1] process to the development of MacXim (Model And Code Xml-based Integrated Meter), an OSS Java tool that extracts static measures from source code [3].

We describe a set of quality and productivity measures that were applied before and after the introduction of the SCRUM methodology in the development of MacXim. We then compare the data collected before and after the introduction of SCRUM, to understand if and how the SCRUM process actually affected the development, and we derive a set of observations on how to apply SCRUM to OSS products.

The experience shows that the introduction of SCRUM did not alter the productivity of our team significantly and the quality of our product. However, we were able to better control the development trend than in the non-SCRUM development period.

The paper is structured as follows. Section 2 states the hypotheses and the objectives of our experience. Section 3 describes the case study and the way we apply the SCRUM process to the development of MacXim. Section 4 reports about the results of our experience by introducing a set of productivity and quality measures, showing the data we collected, and discusses the results we obtained. Section 5 reviews related works in the field of agile methodologies. Finally, Section 6 contains our conclusions and an outline of future work.

2 Development and Research Objectives

MacXim is developed in the context of the activity of the QualiPSo project that is devoted to evaluating the quality of OSS, which, along with a set of other tools, it is used to evaluate the quality of the OSS products. Since we are interested in the quality of OSS products, it was therefore natural to include MacXim in the set of OSS products evaluated. As a consequence, the quality of every release of MacXim was thoroughly measured.

The development of the MacXim tool began in a rather unstructured way, that is, without following a well defined process. So, the measures of the quality of MacXim, along with the measures indicating the speed of the development progress, showed that the situation was not as good as we had hoped. Thus, our development objectives were to find the causes of such unsatisfactory performance and find and use mechanisms that could allow us to improve the development process and the quality of our OSS product.

The analysis of the development practice indicated that the main problem was in the too little amount of coordination among developers, who –also because of a lack of communications, favored by the physical distance among the developers– were not able to achieve a shared vision of the project and failed to act consistently.

It was therefore decided to improve the management of the project by enforcing coordination activities. However, we did not want to put too many constraints on the developers, but just to direct their efforts towards a common and shared target.

To this end, we decide to adopt SCRUM [1], a project management technique that allows developers to achieve a very good trade-off between effective coordination and process agility.

Measures concerning the 'pre-SCRUM' situation were available, so we could use those data as a baseline against which we could compare the results obtained after the introduction of SCRUM. Specifically, these measures could help us carry out a quantitative, objective evaluation of the impact of the introduction of SCRUM in an OSS development context. This evaluation led to three main research goals.

First, we wanted to verify if it is at all possible to switch successfully to SCRUM in an ongoing OSS development process. Our project started out following a different process, so it was unclear whether introducing SCRUM would actually be beneficial to the project.

Second, the applicability of SCRUM to a set of geographically distributed developers had also to be assessed: one of the basic practices of the SCRUM method is to attend daily meetings, which can be clearly problematic to organize if the participants reside far from each other.

Third, we wanted to verify the extent to which SCRUM helps improve the quality of the product and the productivity of the process. This objective could be effectively measured by means of the QualiPSo quality evaluation toolset. Section 4 describes the set of measures used to evaluate the productivity and the quality of MACXIM before and after the introduction of SCRUM.

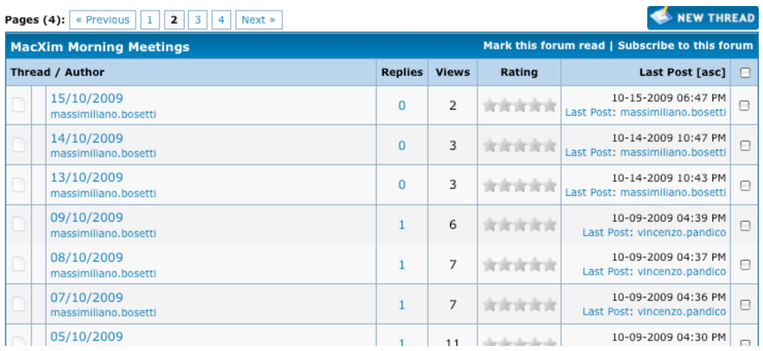
3 The MacXim Case Study

The development of the MacXim prototype started in September 2007 without a clear development methodology in mind. After a few months, the project manager became aware that project requirements and deadlines were difficult to meet without a well planned work and without a dedicated development team. One year later, a dedicated team was formed and MacXim's development started to follow the traditional waterfall process, beginning from high-level requirements and going deep step-by-step. The MacXim team was composed of two junior developers and a project manager until May 2009, when a new junior developer was involved in the project. The team grew from three to four developers in the period June-October 2009, and it lost one developer in October. All the junior developers were involved full-time on MacXim's development, while the project manager is involved only partially (three days per week). Like many OSS development teams, our team was geographically distributed, with developers residing several kilometers away from each other. Problems in meeting deadlines surfaced when the size of team had grew to three people, mainly due to requirements volatility. As a consequence, the project manager recommended the introduction of an Agile methodology to alleviate these problems. As mentioned in

Section 2, we decided to adopt SCRUM because of coordination problems; in any case, we were ready to adapt SCRUM to our specific needs, mainly in order to take into account the distribution of the team. We started the introduction of SCRUM in the middle of July 2009 and we ended it with the V1.0 official release of MacXim at the end of October 2009. The SCRUM master (previously the project manager) suggested that the SCRUM process started with a trial period of two SCRUM sprints, each of two weeks. The trial period succeeded, so we decided to adopt the SCRUM methodology. Each sprint was supplied with a *Sprint Backlog* containing the tasks needed to implement a certain number of features in the Product Backlog. The SCRUM master, in collaboration of the team, assigned each sprint to the developer with the least workload and according to the set priority and the team member skills.

Because of the physical distribution of the MacXim developers, the daily meetings prescribed by SCRUM could not be attended in person. Therefore, we adopted an online-based approach, by using a forum and a videoconferencing system. We substituted morning meetings with an online forum. We structured our forum (see Fig. 1) by creating a thread per day where each developer wrote its comments by replying to three questions:

- What have you completed, with respect to the backlog, since last daily meeting?
- What specific tasks, with respect to the backlog, do you plan to accomplish until the next daily meeting?
- What obstacles got in your way of completing this work?


Pages (4): « Previous 1 2 3 4 Next »
NEW THREAD
MacXim Morning Meetings
Mark this forum read | Subscribe to this forum

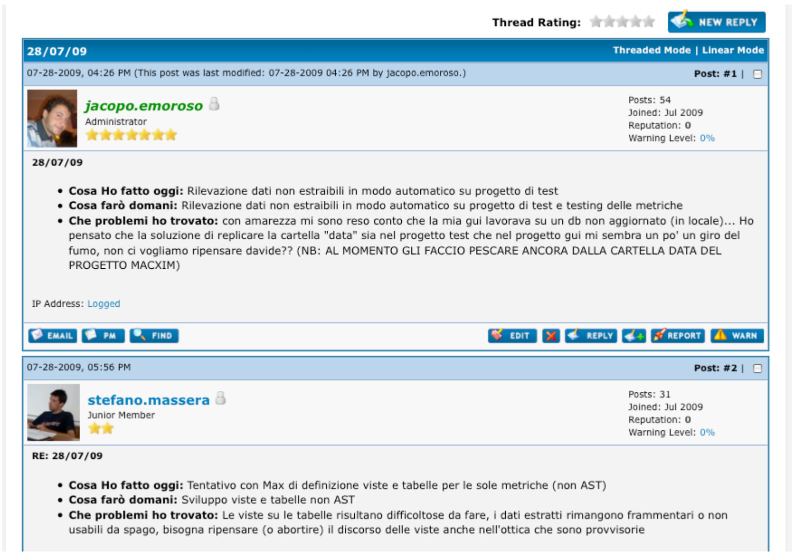
Thread / Author	Replies	Views	Rating	Last Post [asc]
15/10/2009 massimiliano.bosetti	0	2	☆☆☆☆☆	10-15-2009 06:47 PM Last Post: massimiliano.bosetti
14/10/2009 massimiliano.bosetti	0	3	☆☆☆☆☆	10-14-2009 10:47 PM Last Post: massimiliano.bosetti
13/10/2009 massimiliano.bosetti	0	3	☆☆☆☆☆	10-14-2009 10:43 PM Last Post: massimiliano.bosetti
09/10/2009 massimiliano.bosetti	1	6	☆☆☆☆☆	10-09-2009 04:39 PM Last Post: vincenzo.pandico
08/10/2009 massimiliano.bosetti	1	7	☆☆☆☆☆	10-09-2009 04:37 PM Last Post: vincenzo.pandico
07/10/2009 massimiliano.bosetti	1	7	☆☆☆☆☆	10-09-2009 04:36 PM Last Post: vincenzo.pandico
05/10/2009	1	11	☆☆☆☆☆	10-09-2009 04:30 PM

Fig. 1. Morning meeting list

Fig. 2 gives an example of a morning meeting post. The complete forum is available on http://qualipso.dscpi.uninsubria.it/forum (in Italian).

Every morning, each developer had to read all posts and, in case he had ideas on how to solve any posted problem, he was encouraged to suggest the solution.

At first, the team was uneasy about spending time in writing and reading the online morning meeting posts. Anyway, after a while, the team understood the usefulness of these virtual meetings, and both our SCRUM master and our "product owner" (i.e., the QualiPSo manager in charge of tool development) appreciated the possibility of keeping track of the work done on a day by day basis.


Thread Rating: NEW REPLY

28/07/09 Threaded Mode | Linear Mode

07-28-2009, 04:26 PM (This post was last modified: 07-28-2009 04:26 PM by jacopo.emoroso.) Post: #1 |

jacopo.emoroso
Administrator

Posts: 54
Joined: Jul 2009
Reputation: 0
Warning Level: 0%

28/07/09

- **Cosa Ho fatto oggi:** Rilevazione dati non estraibili in modo automatico su progetto di test
- **Cosa farò domani:** Rilevazione dati non estraibili in modo automatico su progetto di test e testing delle metriche
- **Che problemi ho trovato:** con amarezza mi sono reso conto che la mia gui lavorava su un db non aggiornato (in locale)... Ho pensato che la soluzione di replicare la cartella "data" sia nel progetto test che nel progetto gui mi sembra un po' un giro del fumo, non ci vogliamo ripensare davide?? (NB: AL MOMENTO GLI FACCIO PESCARE ANCORA DALLA CARTELLA DATA DEL PROGETTO MACXIM)

IP Address: Logged

EMAIL PM FIND EDIT REPLY REPORT WARN

07-28-2009, 05:56 PM Post: #2 |

stefano.massera
Junior Member

Posts: 31
Joined: Jul 2009
Reputation: 0
Warning Level: 0%

RE: 28/07/09

- **Cosa Ho fatto oggi:** Tentativo con Max di definizione viste e tabelle per le sole metriche (non AST)
- **Cosa farò domani:** Sviluppo viste e tabelle non AST
- **Che problemi ho trovato:** Le viste su le tabelle risultano difficoltose da fare, i dati estratti rimangono frammentari o non usabili da spago, bisogna ripensare (o abortire) il discorso delle viste anche nell'ottica che sono provvisorie

Fig. 2. Morning meeting example

We substituted also in-person sprint meetings with online reports via forum. Our team met online every two weeks: we had four hours meetings in video conference. The team leader was always in charge of writing the sprint retrospective on the forum.

During these meetings, the team began to grow together and show increasing involvement and project knowledge. The team completed a successful sprint, and began to cooperate almost immediately. The most interesting point of this process is that, every time one of our members bumped into a problem, the other developers quickly helped him. Not only did the team members share a common goal, but they actually worked together in an effective manner.

In order to mitigate the possible risks due to lack of direct interaction, we decided to meet in person every month.

4 Empirical Results

4.1 Data Collection

We started collecting data about quality and productivity measures in March 2009 and we ended the data collection at the end of October 2009 with the first official release of MacXim. For two weeks, at the end of June, we stopped the development of MacXim and started a quality and testing campaign to reduce the number of errors and warnings into the code. The campaign was organized both implementing a set of unit test cases and integration test cases, and also running third-party tools such as PMD [20] and Checkstyle [19] to improve code quality and adhere to coding standards. This activity ended at the beginning of July. In the middle of July 2009, we changed the development process and we kept on collecting data in the same way as we did in the previous period.

As fine-grained indicators to assess our initial research objectives (see Section 2), we considered measures for the following product and process attributes.

- Code organization. We expected that a better coordination among developers would lead to better structured code. We used the total number of packages, the average number of effective lines of codes (eLoc) per class and the total number of classes to measure the organization of the code.
- The pace of production. We used the number of classes to measure the "amount of product", as classes are more representative than LOC in an object-oriented development, though sufficiently fine-grained to show quantifiable progress on a weekly basis. In any case, we also take into account the productivity in terms of code produced and we used to eLoc measure to evaluate the productivity.
- Code quality. We used traditional measures for internal qualities, like Chidamber and Kemerer's measures of modularization [3] and McCabe's complexity measure [4], as well as elementary code assessment rules, which provide rather rough evaluations, but are expressive and easy to collect. Also, the amount of comments in the code, which is often a neglected concern, was considered as a quality factor.
- Sprint effectiveness. Finally, as a measure of the effectiveness of the project coordination through SCRUM we counted the percentage of user stories carried out in each sprint period, taking into account the evolution of the development team.

4.2 Case Study Result

At the end of the data collection process, we started the analysis of the data and we derive a set of charts as shown in figures 3, 4, 5, 6, and 7.

Fig. 3 shows the evolution of the measures we used to trace the evolution of project size and code organization. We collected the total number of packages, the average number of eLoc per class and the total number of classes. All these measures have been reported in the figure along with the evolution in the number of team members because of their correlation. Fig. 3 shows that the number of packages and the total number of classes increase after the introduction of SCRUM, while the average number of eLoc per class decreases. The behaviors of the number of packages and of the total number of classes show that MAXCIM's growth pace continued after the introduction of SCRUM, and the code seemed to be well modularized as far as size was concerned, since the average number of eLoc per class appears to be decreasing. Even though these trends seemed to exist even before its introduction, SCRUM was at least not detrimental to size aspects. Moreover, we can observe that the introduction of SCRUM leveled the max and min peaks we had in the previous period for the average eLocPerClass. This is probably due to the general SCRUM practice that suggests frequent intermediate deliveries with working functionality, like all other forms of agile software processes. So, the project can change its requirements according to changing needs, focus on small functionalities, and implement these functionalities into well modularized classes.

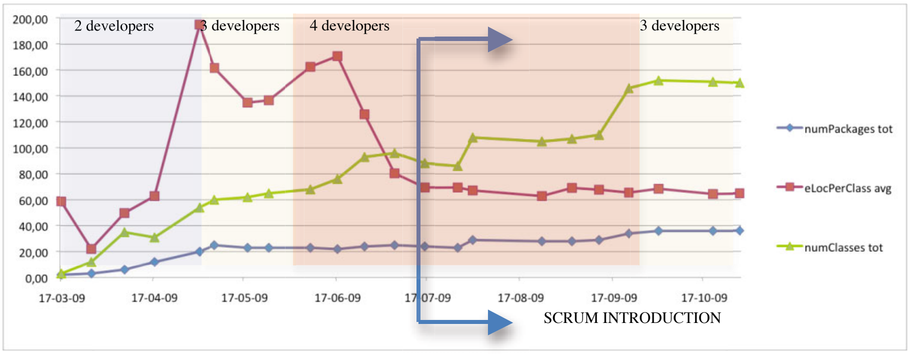


Fig. 3. Size measures over time

Fig. 4 shows the average number of lines of comment per eLoc. A preliminary observation of the chart suggests that this measure decreases after the introduction of SCRUM. This is not actually true because we can observe a growth only during the quality/testing period, which occurred around the end of June and the beginning of July 2009. The average number of lines of comment per eLoc was actually decreasing before that period, between mid-April and mid-June 2009. So, the same decreasing trend was confirmed even after the introduction of SCRUM, in the period from mid-July to mid-October 2009.

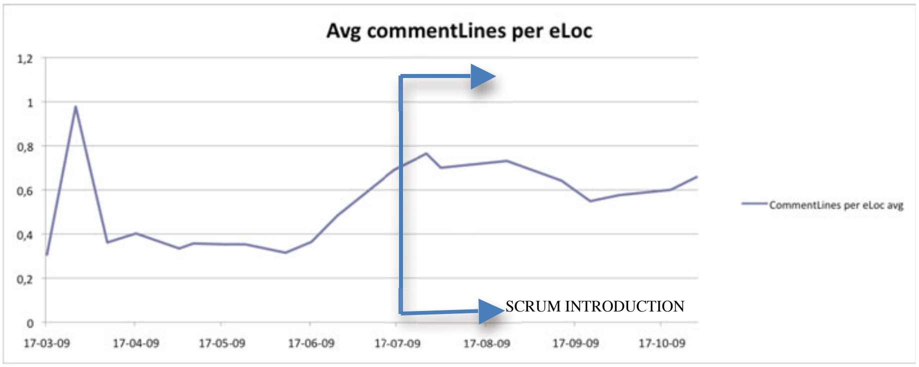


Fig. 4. Comment lines number per eLOC

Fig. 5 reports on code quality measures. We computed the average Lack of Cohesion of Methods metric (LCOM per class), the average Coupling Between Object classes (CBO per class), and the average Cyclomatic Complexity in the methods of each class. As stated in the literature, high values of these measures are undesirable, while low values suggest good internal quality of the object-oriented software product. In our experience, the Cyclomatic Complexity decreases slowly but constantly after the introduction of SCRUM, while the other two measures follow an increasing trend. As for CBO, excessive coupling between object classes is detrimental to modular design and prevents reuse. The more independent a class is, the easier it is to reuse it in another application. The larger the coupling, the higher the sensitivity to changes in other parts of the design, and therefore maintenance is more difficult. This is in contrast with the data we collected in Fig. 3, which seem to suggest an improvement of code modularization and as consequence code maintenance. The increment of CBO may be related to the high increment we had on the total number of classes after the introduction of SCRUM. In any case, the CBO value of MacXim is very low (e.g., a high CBO value is considered greater than 14 [21]). As for LCOM, the quality and testing activities performed just before the adoption of SCRUM strongly reduced LCOM from a value of 10 to a value of 4. This value suddenly increases after the introduction of SCRUM with a peak of value 7 at the end of September 2009. This negative value is occurs at the same time as the max value of the total number of classes, thus suggesting a relationship among the two measures. On average, the LCOM value computed during the Waterfall process is equal to the one computed during the SCRUM process.

In any case, the CBO and the LCOM values seem to suggest that the testing activity performed during the SCRUM process is not as efficient as the one performed during the previous quality/testing activity. This can be explained by the inexperience of our testers. Testers that are not skilled in processes that use iterative lifecycle may encounter greater difficulties when testing within each iteration, rather than at the end of a development lifecycle, deciding what to test when a product is still unfinished, and working with other team members to figure out what to test, rather than testing from requirements documents.

Fig. 6 shows the number of warnings per eLoc computed by means of PMD and Checkstyle. The number of warnings was very high during the Waterfall process (an average of 5 warnings per eLoc), sharply decreased during the quality check period (with the result of 1 warning per eLoc), and remained almost unchanged with a very slow and linear increasing trend (currently, 2 warnings per eLoc) after the introduction of SCRUM.

Fig. 7 shows the rate of successful sprints and the evolution of the developers number in our SCRUM team. With four consolidated developers the success rate

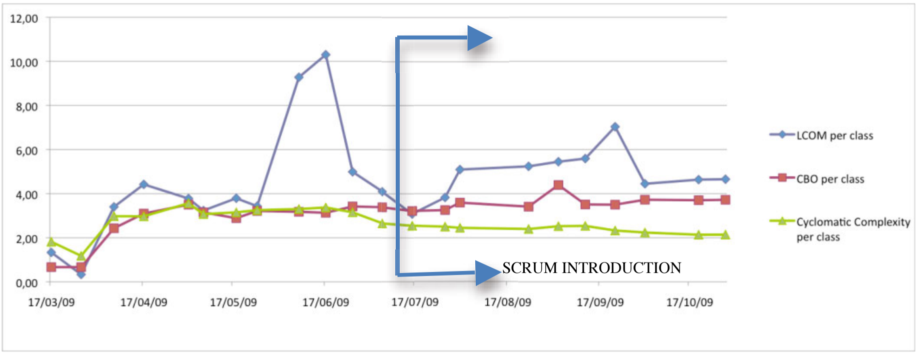


Fig. 5. LCOM, CBO and Cyclomatic Complexity

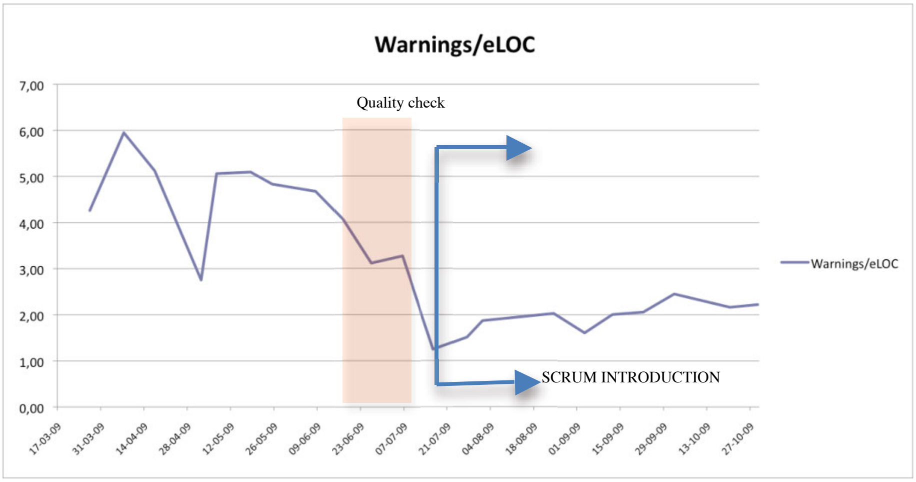


Fig. 6. Coding Standards Warnings / eLOC

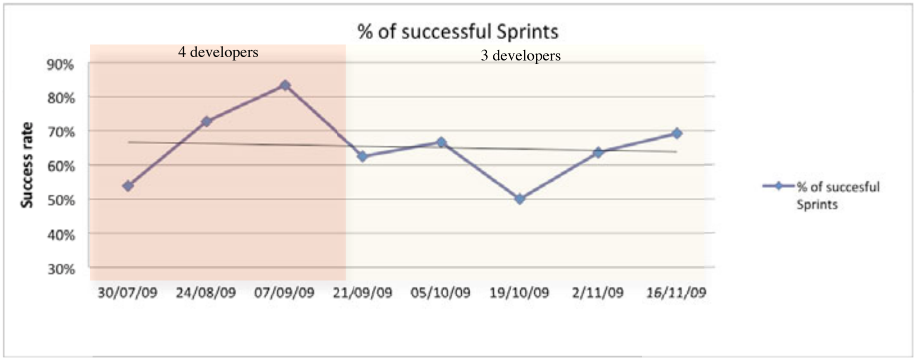


Fig. 7. Percentage of completed sprints

strongly increases from 55% of successful sprints to a rate of 85%. The success rate gets worse with the unexpected renovation of the team at the end of September. Probably, this renovation has invalidated the effort predictions and estimations the team set at the beginning of the SCRUM process. The linear trend of successful sprints (the light grey line in the chart) is almost constant to the rate 65%. Probably, a better definition of sprint backlogs and tasks could increase this rate. Unfortunately, the inherent characteristics of MacXim made difficult to break down its features into tasks of few hours of work.

5 Related Work

The vast majority of SCRUM's practices are not new to SW Engineering. The main idea of SCRUM is basically the same behind the Barry Boehm's Spiral Model [5]. SCRUM speeds up the requirement adaptability of the spiral model.

SCRUM aims at improving the quality of the software development process by means of short time boxes called sprints. In SCRUM, a sprint usually lasts from one to four weeks and a project lasts up to a few months. In this period, as in the spiral model, requirements are expected to change.

There are several studies reporting experiences on success stories in introducing SCRUM into a company but few studied the possibility of introducing SCRUM into a running Open Source process.

The experiment that is most similar to ours is reported in [5]. The adoption of SCRUM in a distributed open source project (PyPy) is reported. In particular, SCRUM was introduced in a project that had been "test driven" and had employed automated test suites and unit tests from the beginning. However, the development environment – with on-site customers, open workspaces and pair programming for each team– was quite different from ours. As in our case, the developers of PyPy were geographically distributed (i.e., several distributed sub-teams collaborate to the project), but they used only sprints to provide an accelerated and collaborative physical practice.

In [10] a two year long industrial case study shows that after the introduction of SCRUM into an existing software development organization, the amount of overtime decreased, allowing developers to work at a more sustainable pace, while, at the same time, customer satisfaction increased. This study differs from ours because of the nature of the team, which was located in a single site: they applied the classical SCRUM process, with in person pair programming and daily meetings.

An interesting work [11] shows the introduction of SCRUM in small teams, where the total number of developers was close to our team's. They found out that small teams can be flexible and adaptable in defining and applying appropriate variant of SCRUM.

Another interesting industrial report [8] shows the introduction of SCRUM in five companies with different sizes and different technologies, with positive results in all cases: SCRUM dramatically improved the communication and the delivery of code.

In general, this paper goes a bit further with respect to previous experience reports, since we do not just report a specific experience in introducing SCRUM. Instead, we suggest a new adaptation of SCRUM that can be used in distributed projects, where daily face-to-face communication is impossible.

6 Conclusions and Future Work

Here is what we may conclude as to the research objectives listed in Section 2. Our experience with MacXim's development suggested that:

1) It is possible to switch successfully to SCRUM in an ongoing (OSS) development process. In effect, the analysis confirms that the introduction of SCRUM does not significantly negatively affect the values of the measures we computed and, in some cases, we had an improvement of these measures, for example for code modularization and Cyclomatic Complexity. Moreover, SCRUM seems to have made productivity more predictable, without min and max peaks, but with smoother curves than the ones we had during the Waterfall development.

2) It is possible to apply SCRUM to a set of geographically distributed developers. As one of the basic practices of the SCRUM method is to attend daily meetings, we replaced face-to-face meetings with online mechanisms of communication. In any case, SCRUM seemed to have significantly improved the communication among the developers.
3) With SCRUM, we obtained a code of high quality, without substantially altering the quality level we reached with the quality check activity, and we were able to meet the cutoff deadline for the first official release of MacXim.

As a final consideration, applying the pure SCRUM methodology to an OSS development community is not possible because of geographical, cultural, and communication problems. Often, developers with different cultures are located all over the world, generally in different time zones. Therefore, it is not possible neither to have morning meetings nor sprint reports together. In our team, we applied a slightly modified version of SCRUM and we substituted in person meetings with an online forum. Moreover, we think that SCRUM can be applied to OSS projects that are characterized by a limited number of contributors to allow a punctual communication among developers.

Currently, we are collecting new SCRUM data by monitoring the development of the next release of MacXim planned for the end of 2010. So, we will be able to compare these new data with the old ones and check whether we can confirm the conclusions we obtained with this first experience. Actually, we expect further process and product improvements as measured by all the measures we collect.

Acknowledgments

The research presented in this paper has been partially funded by the IST project (http://www.qualipso.eu/), sponsored by the EU in the 6th FP (IST-034763); the FIRB project ARTDECO, sponsored by the Italian Ministry of Education and University; and the projects “Elementi metodologici per la descrizione e lo sviluppo di sistemi software basati su modelli” and “La qualità nello sviluppo software,” funded by the Università degli Studi dell’Insubria.

The work reported was made possible by the enthusiast cooperation of the members of the development team: Marco Binda, Massimiliano Bosetti, Jacopo Emoroso, Stefano Massera, Vincenzo Pandico.

References

1. Schwaber, K., Beedle, M.: Agile Software Development with SCRUM. Prentice Hall, Englewood Cliffs (2001)
2. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B.C., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: Manifesto for Agile Software Development, `http://www.agilemanifesto.org/`
3. Crisà, A.F., del Bianco, V., Lavazza, L.: A tool for the measurement, storage, and pre-elaboration of data supporting the release of public datasets. In: González-Barahona, J.M.,

Conklin, M., Robles, G. (eds.) Workshop on Public Data about Software Development (WoPDaSD 2006), Como (2006)

4. Chidamber, S.R., Kemerer, C.F.: A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering 20, 476–493 (1994)
5. McCabe, T.J.: A complexity measure. IEEE Transactions on Software Engineering 2, 300–320 (1976)
6. Boehm, B.W.: A Spiral Model of Software Development and Enhancement. IEEE Computer 21(5), 61–72 (1988)
7. Northover, M., Northover, A., Gruner, S., Kourie, D.G., Boake, A.: Agile software development: A contemporary philosophical perspective. In: ACM International Conference Proceeding Series, vol. 226, pp. 106–115 (2007)
8. Raymond, E.S.: The Cathedral & the Bazaar. O'Reilly, Sebastopol (2001)
9. Sutherland, J.: Agile can scale: Inventing and reinventing scrum in five companies. Cutter IT Journal. Cutter Information Corp. 14(12) (December 2001)
10. Düring, B.: Sprint Driven Development: Agile Methodologies in a Distributed Open Source Project (PyPy). In: Abrahamsson, P., Marchesi, M., Succi, G. (eds.) XP 2006. LNCS, vol. 4044, pp. 191–195. Springer, Heidelberg (2006)
11. Mann, C., Maurer, F.: A Case Study on the Impact of SCRUM on Overtime and Customer Satisfaction. In: Agile Development Conference, pp. 70–79 (2005)
12. Rising, L., Janoff, N.S.: The SCRUM Software Development Process for Small Teams. IEEE Software 17(4), 26–32 (2000)
13. Feller, J., Fitzgerald, B.: A framework analysis of the open source software development paradigm. In: Proceedings of the twenty first International Conference on Information systems (2000)
14. Ghezzi, C., Jazayeri, M., Mandrioli, D.: Fundamentals of Software Engineering, 2nd edn. Pearson, Prentice Hall (2002)
15. Northover, M., Northover, A., Gruner, S., Kourie, D.G., Boake, A.: Agile software development: A contemporary philosophical perspective. In: ACM International Conference Proceeding Series, vol. 226, pp. 106–115 (2007)
16. Mockus, R., Fielding, T., Herbsleb, J.: A case study of OSS development: the apache server. In: Proceedings of the International Conference on Software Engineering (ICSE), pp. 263–272 (2000)
17. O'Reilly, T.: Lessons from Open-Source Software development. Communications of the ACM 42(4) (1999)
18. QualiPSo portal, `http://www.qualipso.eu` (Accessed: December 2009)
19. Checkstyle tool download, `http://checkstyle.sourceforge.net/` (Accessed: December 2009)
20. PMD tool download, `http://pmd.sourceforge.net/` (Accessed: December 2009)
21. Sahraoui, H.A., Godin, R., Miceli, T.: Can Metrics Help Bridging the Gap Between the Improvement of OO Design Quality and Its Automation? In: Proceedings of the International Conference on Software Maintenance, ICSM (2000)