Adapting SCRUM to the Italian Army: Methods and (Open) Tools

Franco Raffaele Cotugno, Angelo Messina

Adapting SCRUM to the Italian Army: Methods and (Open) Tools

Franco Raffaele Cotugno, Angelo Messina
Stato Maggiore dell'Esercito Italiano
Italy

Abstract. Many software-related technologies, including software development methodologies, quality models, etc. have been developed due to the huge software needs of the Department of Defense (DoD) of the United States. Therefore, it is not surprising that the DoD is promoting open source software and agile approaches into the development processes of the defense contractors[1]. The quality of many open source product has been demonstrated to be comparable to the close source ones and in many cases even higher and the effectiveness of agile approaches has been demonstrated in many industrial settings. Moreover, the availability of the source code makes open source products attractive for obvious reasons (e.g., security, long term maintenance, etc.). Following this trend, also the Italian Army has started using open source software and promotes its usage into the development processes of its contractors, also promoting agile approaches in many contexts focusing on the SCRUM methodology. This paper provides an overview of the SCRUM development process adopted by the Italian Army for the development of software systems using open source technologies.

1 Introduction

Software systems are becoming larger and larger requiring an increasing amount of resources in terms of effort and budget. In particular, in the military environment where the reliability of systems is of paramount importance, software costs and the length of the development cycles represent rising challenges. To reduce costs and development cycles, Open Source Software (OSS) and Agile Methods (AMs) are opportunities that can be investigated even is such environment.

Considering the development process, AMs have been demonstrated their ability in delivering quality software on time focusing on the value from the point of view of the customer [7, 8, 9, 20, 24, 27, 28]. For this reason, in many cases, AMs are able to satisfy the customer needs and create a strong and trusted relationship with the development

[1] DoD Open Source Software (OSS) FAQ available at: http://dodcio.defense.gov/OpenSourceSoftwareFAQ.aspx

team [15, 29, 30, 31, 32]. In particular, SCURM have been selected in many contexts for its focus on agile project management [16].

Traditionally, OSS is characterized by informal development processes, unstable teams, and different levels of quality [10, 11, 12], therefore adopting OSS may be considered risky. However, in many cases, OSS development is supported by software companies (both large and small) and OSS projects are no more interesting only to enthusiasts and volunteers [6]. These aspects increase the interest in a business use of OSS.

Software companies often use different popular models such as the Capability Maturity Model (CMMI) [1] to assess the level of quality of their development process. However, OSS projects can hardly adopt CMMI, because it is too complex, requires extensive efforts to be used, and does not take into account the characteristics of OSS. For these reasons simpler and more suitable models appeared in the last few years such as the QualiPSo OMM [14], QualiPSo MOST [18], OSMM [2, 3], QSOS [4], OpenBRR [5], etc. Such models focus on different aspects of the production: QualiPSo OMM deals with the quality of the production process, while all the others focus mainly on the quality of the final product. Such models can be used to evaluate OSS and decide if such products can be used in different contexts.

The rest of the paper is organized as follows: Section 2 presents some related work; Section 3 discusses how AMs and OSS can be used in the military environment; finally, Section 4 draws the conclusions and presents future work.

2 Related Work

AMs and SCRUM, in particular, have been demonstrated to be very successful in defense projects [17] helping the military to manage in a new way complex projects that need to adapt quickly to continuously changing environments. AMs are used to address such problems allowing developers to ship working versions of the needed systems reducing the costs and delivering quickly. Moreover, there are several approaches to track the status of the development process and the quality of the produced software [21, 22, 25].

During the last 10 years, several open source assessment methodologies have been proposed. Most of them focus mainly on the assessment of the product and do not offer enough metrics to obtain a detailed assessment of the quality of the development process [13, 26].

The first proposed assessment methodology is the Open Source Maturity Model (OSMM) proposed by Cap Gemini [2]. It is the first attempt to standardize the usually ad-hoc assessment approaches of open source projects. The methodology allows the evaluation of 12 product related characteristics and 15 user related characteristics. This way the authors of the methodology allowed a personalized assessment approach based on specific user needs of the FLOSS product.

One year later Navica proposed its own assessment method also called the Open Source Maturity Model (OSMM) [3]. This method is more compact than the one proposed by Cap Gemini. Navica's method is more interesting for our research because

it contains some elements that allow a partial assessment of the development process. The method requires the assessment of six project related characteristics. The following four requires some information about the development process: support, documentation, training, and professional services.

In 2004, the Qualification and Selection of Open Source Software (QSOS) [4] method was proposed by a group of OSS developers, users, and enthusiasts. This method has been released with an open source license, allowing an open adoption without any restrictions. The methodology allows an iterative assessment approach. During the first iteration, the number of tools to evaluate are restricted, during the second iteration, their number is additionally limited and so on, until we reach the last stage where we obtain the tool that scores best.

Open Business Readiness Rating (OpenBRR) [5] was specifically designed to allow the assessment of OSS tools that are mature enough to be used by industry. The method was proposed in 2005 and its authors reused many elements introduced in previous OSS assessment methods. The OpenBRR method contains also a few process-oriented elements such as: quality, security, support, documentation, adoption, community, professionalism, and others.

QualiPSo OMM [14] and QualiPSo MOST [18, 19] have been proposed by the QualiPSo consortium in 2009 as two complementary methodologies to assess both the development process (OMM) and the product (MOST). QualiPSo OMM has been designed to be compatible with CMMI allowing companies already using CMMI to implement OMM in an easy way. Moreover, it focuses on different aspects of the development process analyzing it from three points of view: users (consumers of OSS), developers (producers of OSS), and system integrators. Based on such profiles, the methodology provides customized sets of characteristics that require evaluation. QualiPSo MOST has been designed in conjunction to the QualiPSo OMM methodology but focusing on the assessment of the product quality. Most of the assessment criteria defined in MOST can be automated, simplifying the assessment process.

3 Assessing quality through QualiPSo OMM and MOST

The idea and the structure of OMM and MOST are inspired by the CMMI even if they have been designed specifically to be lightweight and focused on OSS development.

OMM defines three maturity levels (basic, intermediate, and advanced). Each level includes a set of characteristics that need to be assessed. The characteristics are the following:

1. Product Documentation (PDOC)
2. Popularity of the Software Product (REP)
3. Use of Established and Widespread Standards (STD)
4. Availability and Use of a (product) Roadmap (RDMP)
5. Quality of Test Plan (QTP)

6. Relationship between Stakeholders (Users, Developers, etc.) (STK)
7. Licenses (LCS)
8. Technical Environment (Tools, OS, Programming Language, Development Environment) (ENV)
9. Number of Commits and Bug Reports (DFCT)
10. Maintainability and Stability (MST)
11. Contribution to FLOSS Product from SW Companies (CONT)
12. Results of Assessment of the Product by 3rd Party Companies (RASM)

OMM is organized in three levels. Each level includes the characteristics at the lower levels.

The basic level includes:

- Product Documentation (PDOC)
- Use of Established and Widespread Standards (STD)
- Quality of Test Plan (QTP)
- Contribution to FLOSS Product from SW Companies (CONT)
- Licenses (LCS)
- Technical Environment (Tools, OS, Programming Language, Dev Environment.) (ENV)
- Number of Commits and Bug Reports (DFCT)
- Maintainability and Stability (MST)

The intermediate level adds:

- Popularity of the Software Product (REP)
- Availability and Use of a (product) Roadmap (RDMP)
- Relationship between Stakeholders (Users, Developers, etc.) (STK)
- Results of Assessment of the Product by 3rd Party Companies (RASM)

The advanced level includes all the characteristics identified in the basic and intermediate level and adds deepness requiring more details for each characteristic.

The model has been designed to be used in different industrial context from the points of view of the users, the integrators, and the developers. Therefore, it can be used in almost any situation in which OSS is needed [14, 23].

As a complementary methodology, MOST provides the tools to access the quality of the code of OSS. MOST have been designed with the same goals and methodologies of OMM and defines a specific set of characteristics for the evaluation of open source code. [TODO]

3 AMs and OSS in the military environment

AMs are designed to support changing requirements through the implementation of an incremental development approach and the close interaction with the customer. In these approaches, the customer becomes part of the development team allowing a better understanding of the problem and a faster development of a system that is able to satisfy user requirements. In the military environment, software systems are very complex and applying AMs is a challenge. Among the available AMs, SCRUM seems to fit better the environment due to its focus on project management with changing requirements.

SCRUM divides the project activities into short iterations named *sprints* that are able to produce a deployable (even if partial) system able to be fully tested by the customer to provide fast feedback and fix the potential problems or requirements misunderstandings.

A SCRUM team includes several actors: a Product Owner (PO), a SCRUM Master (SM), and a Development Team (DT). To adopt SCRUM to develop software systems through a close collaboration between the Italian Army and external contractors, the SCRUM team should be organized in the following way:

- **Product Owner:** he provides the vision of the final product to the team. He guides the development defining the requirements of the product and the related priorities. Therefore, he is the main responsible for the success or failure of the project. His availability to answer questions from the DT and the SM has a significant impact on the development since he is the keeper of the knowledge about the final product. For these reasons, he needs to be a member of the Army with a wide knowledge about the system under development and a clear vision on the expected results. Moreover, he should be able to involve in the testing of the system the people that will use the final system in real operations. Due to the complexity of the role and the wide knowledge needed, the PO is supported by a specific Support Team that includes people from the Army and people from the contractors with deep knowledge about the systems under development and the adopted technologies. The composition of such Support Team will change based on the specific challenges of each sprint. The main role of this team is to help the PO in the definition of the functional and non-functional requirements (including security issues, compliance to standards, usability, performance, etc.). The contribution of the Support Team is very important in particular at the beginning of the project during the definition of the Product Backlog and during the re-prioritization activities required in case of relevant changes in the Backlog. Moreover, in specific conditions related to the technical complexity of a sprint, the Army personnel of the Support Team could take the role of PO just for the duration of the specific sprint. In this way, the PO will be able to guide the development of a large and complex system that can be hardly managed by a single person and require a wide set of skills and knowledge at different levels of granularity. The PO participates actively in the team in the development and in promoting team building activities in conjunction with the SM.
- **SCRUM Master:** he is an expert in the usage of the SCRUM methodology and has a central role in the SCRUM Team coaching the DT and helping in

the management of the Product Backlog. The SM helps the DT to address problems in the usage and adaptation of the SCRUM methodology to the specific context in which the team works. Moreover, he leads the continuous improvement that is expected from any agile team. An additional duty of the SM is to protect the DT from any external interferences and lead the team in the removal of obstacles that may prevent the team to become more productive. The SM is selected among the personnel of the IT department of the Army. The SM and the PO have to collaborate closely and continuously to clarify requirements and get feedback. In the Army implementation of SCRUM, the SM has a stronger role focusing on supporting the team in focusing on the requirements and priorities defined by the PO.

- **Development Team:** as any SCRUM development team, people belonging to the team are together responsible for the development of the entire product and there are no specialists focusing on limited areas such as design, testing, development, etc. All the members of the team contribute to the entire production. The DT is self-organized and decides without any external constraints how to organize and execute the Sprint Backlog (the selected part of the Product Backlog that has been identified to be executed in a sprint) based on the priorities defined by the PO on the Product Backlog. The DT usually includes between 4 and 10 people, all expert software developers. In this case, the team includes people form the Army and the contractors, even if the contribution of the contractors is predominant. In the pilot project, the team includes 4 people from a contractor and 2 people from the Army, in subsequent projects we expect to be able to replicate such unit to manage more sprints in parallel.

The SCRUM methodology is not a one-size-fits-all approach in which all the details of the process are pre-defined and the development team have to stick with it without any modification. On the contrary, SCRUM defines a high-level approach and a state of mind for the team promoting change management and flexibility in the work organization aimed at satisfying the customer needs. As all the other AMs, SCRUM defines values, principles, and practices focused on close collaboration, knowledge sharing, fast feedback, tasks automation, etc.

The SCRUM development process starts from a vision of the PO. Such vision is a wide and high-level definition of the problem to address that will be refined and narrowed during the development through the Backlog Grooming. The Backlog Grooming is an activity that takes place during the entire development process focusing on sharpening the problem definition, pruning redundant and/or obsolete requirements, and prioritizing the Backlog. Moreover, in this phase, the PO defines the scenarios and the criteria used to test the (and accept) the user stories.

Ones defined the Product Backlog, during the Planning Meeting, the PO and the customers/stakeholders define the detailed user stories assigning them a priority that are used to organize the single sprints. During the sprint planning phase, the PO and the DT agree on the objectives of the sprint. The DT decompose features in the Backlog into detailed tasks. Such list of tasks becomes the Sprint Backlog that will be implemented during the sprint execution producing the (incremental) shippable product. At the end of each sprint, a sprint review takes place to verify the progress of the project comparing it to the expectations. The results of the review are used to adapt

the Product Backlog modifying, removing, and adding requirements. This activity is very important for the success of a project and involves all the member of the SCRUM Team and the interested stakeholders. Additionally, other SCRUM Teams could join the meeting if the project requires the collaboration of more teams. This activity is very important to align the teams with the stakeholders and to manage the project properly even if changes happen. The Product Backlog is dynamic, changing continuously collecting all the requirements, issues, ideas, etc. and should not be considered in a static way, it is designed to support the variability of the environment.

After the Sprint Review, there is the Sprint Retrospective in which the PO, the SM, and the DT analyze together the sprint just finished to evaluate its effectiveness and to identify problems and opportunities to improve.

In the specific context of the Army, the SCRUM Team should be organized in the following way:

1. The PO defines the Product Backlog that, based on the specific objective of the sprint, may include:
    a. High-level views (e.g., Operational Views) describing the functional properties from the operational point of view that can be easily shared and discussed with the operational units.
    b. A natural language description of the user stories of the items of the Product Backlog
2. Based on the Backlog, the PO identify the proper people to include in the Support Team. At this level, only candidate people are identified but no actual assignment is performed because the assignment is performed at each sprint based on the specific competences needed.
3. Once defined the Backlog, the DT defines the Sprint Backlog and the PO assign specific people to the Support Team selecting them from the pool identified at the beginning and satisfying the needs of the current sprint.
4. After these activities, the DT can start the execution of the sprint. Once the development activities are completed (including all the functional and non-functional testing), the current design documents can be generated automatically through tools for code reverse engineering. Such documentation will be used at the beginning of the subsequent sprint.
5. At the end of the execution, a review is performed involving the entire SCRUM team and the stakeholders (mainly from the operational units) able to assess the usability of the system in the real context.
6. At the end of the sprint, a retrospective takes place involving the PO, the SM, and the DT. Such activity focuses on improving the effectiveness of the team through process improvement strategies and evaluating new approaches.

In the context of the Army, testing has a very important role due to several constraints:

- **Testing at development level:** testing activities performed by the development team in the development environment and in a specific testing

environment with the same characteristics of the deployment environment. Such activities are the traditional ones performed in any Agile team.

- **Testing at operating unit level:** testing activities performed by the operating units in limited (but real) environments to verify the actual effectiveness of the developed systems in training and real operating environments.
- **Testing at certification level:** testing activities performed to verify the compliance of the developed systems to the national and international (NATO) standards that are needed for authorizing the usage of the systems in national and joint NATO operations.

4 Conclusions and future work

In this paper, we have presented an overview of the usage of QualiPSo OMM and MOST for the evaluation of the quality of OSS and SCRUM for addressing the development needs of the Italian Army. This is just an initial step for providing a comprehensive analysis on how OSS and AMs can be used in such environment reducing costs and increasing the effectiveness of the development teams.

References

1. Amoroso E., Watson J., Marietta M., Weiss J. (1994) A process-oriented methodology for assessing and improving software trustworthiness, 2nd ACM Conference on Computer and communications security.
2. Duijnhouwer F.-W., Widdows C. (2003) Capgemini Expert Letter Open Source Maturity Model, Capgemini.
3. Goldman R., Gabriel R. P. (2005) Innovation Happens Elsewhere - Open Source as Business Strategy Boston, Morgan Kaufmann.
4. Atos Origin, Method for Qualification and Selection of Open Source Software (QSOS), http://www.qsos.org
5. Wasserman, A., Pal, M., Chan, C. (2005) Business Readiness Rating Project, BRR Whitepaper, http://www.openbrr.org/wiki/images/d/da/BRR_whitepaper_2005RFC1.pdf
6. Dueñas C.J., Parada H.A., Cuadrado G. F., Santillán M., Ruiz J.L. (2007) Apache and Eclipse: Comparing Open Source Project Incubators, IEEE Software, vol.24, no.6, Nov/Dec.
7. Sillitti A., Succi G. (2005) Requirements Engineering for Agile Methods, in Engineering and Managing Software Requirements (Eds.: Aurum A., Wohlin C.), Springe
8. Janes A., Remencius T., Sillitti A., Succi G. (2013) Managing Changes in Requirements: an Empirical Investigation, Journal of Software: Evolution and Process, Wiley, Vol. 25, No. 12, pp. 1273-1283, December.
9. Di Bella E., Fronza I., Phaphoom N., Sillitti A., Succi G., Vlasenko J. (2013) Pair Programming and Software Defects – a large, industrial case study, Transaction on Software Engineering, IEEE, Vol. 39, No. 7, pp. 930 - 953, July
10. Di Bella E., Sillitti A., Succi G. (2013) A multivariate classification of open source developers, Information Sciences, Elsevier, Vol. 221, pp. 72 - 83, February.

11. Scotto M., Sillitti A., Succi G. (2007) Open Source Development Process: a Review, International Journal of Software Engineering and Knowledge Engineering, World Scientific, Vol. 17, No. 2, pp. 231 - 248, April.
12. Jermakovics A., Sillitti A., Succi G. (2013) Exploring collaboration networks in open-source projects, 9th International Conference on Open Source Systems (OSS 2013), Koper, Slovenia, 25 - 28 June.
13. Petrinja E., Sillitti A., Succi G. (2010) Comparing OpenBRR, QSOS, and OMM Assessment Models, 6th International Conference on Open Source Systems (OSS 2010), Notre Dame, IN, USA, 30 May - 2 June.
14. Petrinja E., Nambakam R., Sillitti A. (2009) Introducing the Open Maturity Model, 2nd Emerging Trends in FLOSS Research and Development Workshop at ICSE 2009, Vancouver, BC, Canada, 18 May.
15. Sillitti A., Ceschi M., Russo B., Succi G. (2005) Managing Uncertainty in Requirements: a Survey in Plan-Based and Agile Companies, 11th IEEE International Software Metrics Symposium (METRICS 2005), Como, Italy, 19 - 22 September.
16. Schwaber K. (2004) Agile Project Management with Scrum, Microsoft Press.
17. Crowe P., Cloutier R. (2009) Evolutionary Capabilities Developed and Fielded in Nine Months, CrossTalk: The Journal of Defense Software Engineering, Vol. 22, No. 4, pp 15-17, May/June.
18. Del Bianco V., Lavazza L., Morasca S., Taibi D., Tosi D. (2010) An investigation of the users' perception of OSS quality, 6th International Conference on Open Source Systems (OSS 2010), Notre Dame, IN, USA, 30 May - 2 June
19. Lavazza L., Morasca S., Taibi D., Tosi D. (2012) An empirical investigation of perceived reliability of open source java programs, 27th Symposium On Applied Computing (SAC 2012), Riva del Garda, Italy.
20. Moser R., Abrahamsson P., Pedrycz W., Sillitti A., Succi G. (2007) A case study on the impact of refactoring on quality and productivity in an agile team, 2nd IFIP Central and East European Conference on Software Engineering Techniques (CEE-SET 2007), Poznań, Poland, 10 - 12 October.
21. Scotto M., Sillitti A., Succi G., Vernazza T. (2006) A Non-Invasive Approach to Product Metrics Collection, Journal of Systems Architecture, Elsevier, Vol. 52, No. 11, pp. 668 - 675, Novembe.
22. Scotto M., Sillitti A., Succi G., Vernazza T. (2004) A Relational Approach to Software Metrics, 19th ACM Symposium on Applied Computing (SAC 2004), Nicosia, Cyprus, 14 - 17 March.
23. Clark J., Clarke C., De Panfilis S., Granatella G., Predonzani P., Sillitti A., Succi G., Vernazza T. (2004) Selecting Components in Large COTS Repositories, Journal of Systems and Software, Elsevier, Vol. 73, No. 2, pp. 323 - 331, October.
24. Coman I., Sillitti A., Succi G. (2008) Investigating the Usefulness of Pair-Programming in a Mature Agile Team, 9th International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP2008), Limerick, Ireland, 10 - 14 June.
25. Sillitti A., Janes A., Succi G., Vernazza T. (2004) Measures for Mobile Users: an Architecture, Journal of Systems Architecture, Elsevier, Vol. 50, No. 7, pp. 393 - 405, July.
26. Kovács G.L., Drozdik S., Succi G., Zuliani P. (2004) Open source software for the public administration, 6th International Workshop on Computer Science and Information Technologies.
27. Fronza I., Sillitti A., Succi G. (2009) Modeling Spontaneous Pair Programming when New Developers Join a Team, 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 2009), Lake Buena Vista, FL, USA, 15 - 16 October.
28. Coman I., Robillard P. N., Sillitti A., Succi G. (2014) Cooperation, Collaboration and Pair-Programming: Field Studies on Back-up Behavior, Journal of Systems and Software, Elsevier.

29. Janes A., Remencius T., Sillitti A., Succi G. (2013) Managing Changes in Requirements: an Empirical Investigation, Journal of Software: Evolution and Process, Wiley, Vol. 25, No. 12, pp. 1273 - 1283, December.
30. Fronza I., Sillitti A., Succi G., Vlasenko J., Terho M. (2013) Failure Prediction based on Log Files Using Random Indexing and Support Vector Machines, Journal of Systems and Software, Elsevier, Vol. 86, No. 1, pp. 2 - 11, January.
31. Janes A., Succi G. (2012) The dark side of agile software development, ACM international symposium on New ideas, new paradigms, and reflections on programming and softwar, Onward! '12.
32. Sillitti A., Succi G., Vlasenko J. (2012) Understanding the Impact of Pair Programming on Developers Attention: A Case Study on a Large Industrial Experimentation", 34th International Conference on Software Engineering (ICSE 2012), Zurich, Switzerland, 2 – 9 June.