Assessing the impact of Scrum in airborne software quality assurance

Fernanda Comenale Portugal Motta Fernandes[1], Ricardo Bedin França[1]

[1]EMBRAER S.A.
Rod. Presidente Dutra, km 134 – 12247-820 – São José dos Campos – SP – Brazil
{fernanda.fernandes,ricardo.franca}@embraer.com.br

Abstract. *In this paper, we present an evaluation of Scrum's impacts over the software quality assurance process in the embedded aerospace software domain. We present basic concepts of the DO-178C and Scrum, then we assess related work and introduce the case study used as reference for our study, as well as the perceived results of using Scrum in both the development teams and in quality assurance. Finally, we summarize good practices for using Scrum in software quality assurance teams.*

Resumo. *Este artigo apresenta uma avaliação dos impactos do Scrum no processo de garantia de qualidade de software no domínio de software embarcado aeronáutico. Após uma introdução ao documento DO-178C e ao Scrum, trabalhos correlatos são avaliados e o caso de estudo utilizado neste artigo é apresentado, bem como os resultados do uso de Scrum tanto nos times de desenvolvimento quanto na garantia de qualidade. Por fim, sugerem-se boas práticas para o uso de Scrum em times de garantia de qualidade.*

1. Introduction

1.1. Context

Software quality assurance (SQA[1]), in the domain of embedded airborne software products, is a peculiar job. Its duties, according to the DO-178C guidance [SC-205 RTCA 2011], go beyond traditional product quality concepts (e.g. being bug-free) and, in addition, deals with the software development process itself. Therefore, SQA teams have different structure and workflow from "pure" development groups. SQA engineers do not develop embedded code for the projects they work on - actually, SQA teams are independent from development teams and have their own activities, which often involve verifying and enforcing process compliance. However, SQA workflow is influenced by the organization of development teams: the depth and frequency of activities such as reviews and audits depend on the development cycle adopted by each team.

In particular, if developers work with agile, iterative development methods (e.g. Scrum), quality assurance activities shall be synchronized with the product readiness attained with such iterations - this often implies a need for agile practices in SQA teams as well. In this work, we tackle the implications of Scrum in SQA activities.

[1] In this paper, we use SQA when referring to software quality assurance as defined by the DO-178C.

1.2. Objectives

The general objective of this work is to evaluate the impact of agile development practices in airborne software quality assurance teams. Specifically, we intend to:

- **Assess the influence of agile development teams over SQA activities**: Although many agile practices are available for software development, we shall focus on Scrum, as it often dictates product development rhythm. SQA strategies depend on this rhythm to be effective.
- **Identify strengths and challenges in the use of Scrum practices by SQA teams:** Since we assume iterative and incremental product development, it seems natural to perform SQA activities iteratively and incrementally, too. Here, too, we focus on Scrum, as coding-related agile practices have no use in SQA.
- **Propose adaptations in Scrum practices to make them more suitable to SQA:** Since Scrum is not a rigid method, it can be tuned according to the team needs and, in our study, we intend to use this freedom to make it more suitable to SQA activities.

1.3. Research method

The two first specific objectives will be accomplished primarily with the help of a case study, consisting of a SQA team that has had experience with Scrum. Qualitative assessments will be obtained by discussions and interviews with team members, whereas quantitative ones will be based on team performance indicators.

Proposed adaptations in Scrum for SQA will be based both in good practices already used by the case study SQA team and in solutions devised by the authors in conjunction with the case study team.

1.4 Structure of this paper

We follow this introduction with the main concepts involved in this work: section 2 describes the DO-178C document - with emphasis on its considerations on software quality assurance - and section 3 presents the Scrum agile method. Related work is presented and compared to ours in section 4, while the case study itself and its results are shown in section 5. We draw conclusions and present future work in section 6.

2. Quality Assurance in DO-178C

Aircraft software programs may be among the most critical of all software products: for certain aircraft systems, a software glitch might trigger a system failure condition that could put entire aircraft - and all souls on board - in jeopardy. In order to reduce such risk, the development of embedded airborne software follows "heavyweight" processes, typically guided by the DO-178C document: aviation certification authorities (such as FAA[2] in the United States, EASA[3] in Europe and ANAC[4] in Brazil) do not explicitly

[2] http://www.faa.gov/

[3] https://www.easa.europa.eu/

[4] http:// www.anac.gov.br

mandate its use, but they usually acknowledge the DO-178C as the main guidance to evaluate the soundness and rigor of an airborne software product life cycle process.

The DO-178C provides extensive guidance for the entire life cycle of an airborne software product: as shown in Figure 1, it comprises "life cycle processes" which encompass the development itself (requirements, design, coding and integration) and "integral processes" that exist throughout the development, such as verification, configuration management, certification liaison and SQA itself. The basis for all product life cycle processes is the software planning process, which "produces the software plans and standards that direct the software development processes and the integral processes" [SC-205 RTCA 2011].

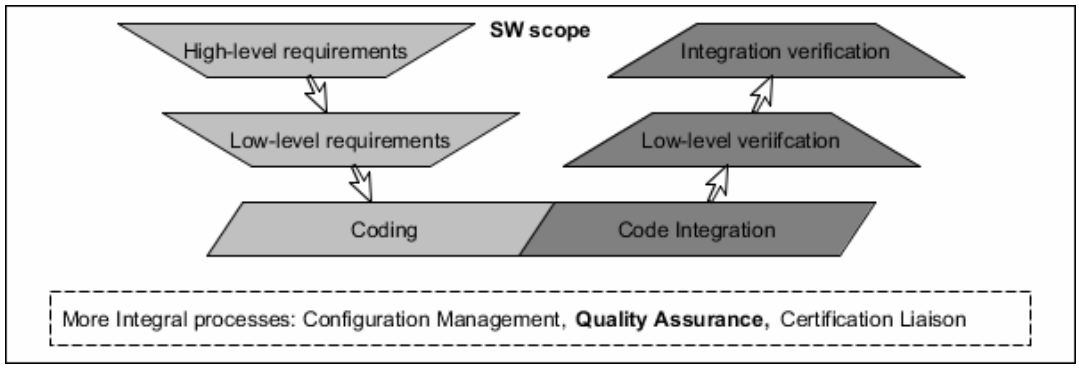

Figure 1. A software life cycle scheme based on the DO-178C

The concept of SQA in DO-178C emphasizes process quality: SQA teams shall make sure that development teams are compliant to the plans and standards defined in their project. Whereas Rierson [2013] remarks that SQA activities may go beyond the process and encompass the product itself, our work shall focus on process quality assurance. Typical SQA activities include auditing life cycle artifacts, witnessing development and verification activities (e.g. code builds and test procedures) and reviewing product plans and standards.

SQA teams shall be *independent* from development teams to meet DO-178C guidance. In this context, independence is achieved when two conditions are met:

- The person that has developed an artifact (e.g. a plan) and the person that performs SQA activities over that artifact cannot be the same.
- The SQA team shall have authority to enforce corrective actions when it judges them necessary.

3. Scrum basics

Scrum is a well-known agile technique that focuses on software management [Gomes 2013] rather than development issues. It aims to provide more adaptability to a dynamic environment (e.g. clients who frequently change their requirements) by making use of short development iterations (called "sprints") and frequent deliveries to product customers. Developers may fulfill the following roles in Scrum:

- **Product Owner** – As implied by the name, the Product Owner is the main product stakeholder. The product owner manages the *Product Backlog* - a list of

activities (usually, software requirements that become functionality) that are needed to implement and deliver the software product. Those activities are prioritized and their execution times are estimated before their start.

- **Scrum Master** – The Scrum Master usually has two main roles: enforcing correct Scrum practices in the team and removing any impediments that may disturb the team activities during the sprint.
- **Scrum Team** – The Scrum Team performs the sprint activities. Scrum teams are usually small [Schwaber and Sunderland 2013] (between five and nine people) and team members must be well-trained and cross-functional, in order to operate efficiently and with a high level of empowerment.

Every sprint commences with a *Sprint Planning*, when top-priority activities from the Product Backlog are transferred to the *Sprint Backlog* in order to be performed in that specific sprint. The Scrum Team keeps up to date with activity status and possible impediments by means of *daily Scrums* that usually last about 15 minutes [Schwaber and Sunderland 2013].

When a sprint ends, the team produces a potentially shippable product increment — i.e. product increment with value to customer, and results are presented to the Product Owner in the *Sprint Review*. The sprint itself is evaluated in a *Sprint Retrospective*, which may yield improvement proposals. The Scrum workflow is illustrated in Figure 2.

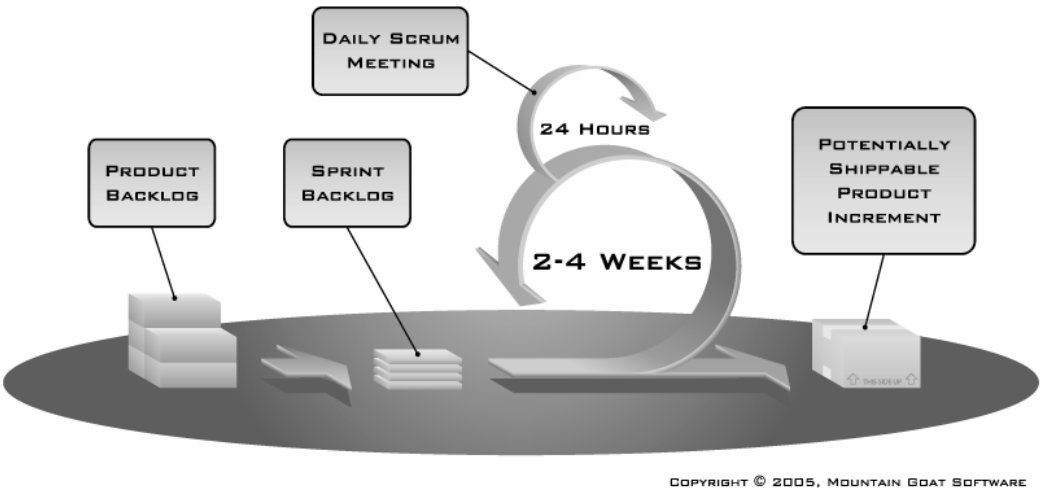

Figure 2 - Scrum workflow (source: Mountain Goat Software)

4. Related Work

The coupling of software quality assurance - as described in DO-178C - and agile practices is, to the best of our knowledge, a relatively unexplored topic. There is a fair amount of work related to agile practices and software quality but, as stated in the introduction, they tackle product quality rather than process quality: Itkonen et al [2005] focus their work on agile testing, while Ambler [2005] underlines several development and techniques but also focus on product quality. Stamelos and Sfetsos [2007] devote their book to several quality assurance topics but they are also product-related.

There are previous works that evaluate the use of agile practices in software products subject to DO-178C guidance: Chisholm's dissertation [Chisholm 2007] assesses the use of agile practices through a whole product life cycle, based on the DO-178B (the predecessor of DO-178C) practices. However, quality assurance activities are a minor part of his study, due to the large number of product-related activities and artifacts involved in the product life cycle. Since agile practices are most easily seen as product-oriented (precluding documentation writing if it adds little or no value), one is tempted to interpret process quality assurance activities as intrinsically "anti-agile", as they often involve a large number of documents that do not add immediate value to the product - although, of course, such documentation has paramount importance when it comes to demonstrate the soundness of all software development processes.

Other works that defend the use of agile methods in airborne software development [VanderLeest and Buter 2009] [Wils et al 2006] [Chenu 2009] focus on development and testing but do not provide specific guidance about the role and workflow of SQA teams in an agile environment.

In this work, we focus on a seemingly neglected (at least, from an "agile" point of view) process that differs significantly from product development and testing: since process quality assurance is mandatory as per DO-178C guidance, performing it efficiently is important to achieve an overall lean development.

5. The effects of SCRUM over SQA

5.1. The case study

The case study used in this paper consists of a SQA team that enforces quality assurance over several software development projects that use Scrum. The most important tasks performed by the team usually are:

- Reviewing plans and standards to assure they are consistent and they meet DO-178C guidance.
- Auditing life cycle artifacts and development environment to evaluate their compliance to plans and standards.
- Witnessing development activities to ensure that work instructions and procedures are being followed correctly.
- Attest, in the closure of project, that all software life cycle data produced during the software life cycle process meet DO-178C guidance.

The SQA team itself decided to adopt (and adapt) Scrum practices and has performed more than fifty sprints since then. The team consists of one product owner and eight team members, including one Scrum Master. SQA product backlog and deliverables are prepared according to development team schedules - priority setting and small adjustments are then made during sprint planning. Daily meetings can be held via conference calls when the team is split in several physical locations and all sprint artifacts are managed with the Jira[5] tool, thus being available to all team members.

[5] https://www.atlassian.com/software/jira

5.2. Enforcing SQA in agile development teams

Rierson [2013] emphasizes the need for SQA during the whole product life cycle - as opposed to trying to enforce compliance only in late development phases. Thus, SQA activities are strongly tied to the schedule and milestones defined by development teams - in fact, one may consider product developers as SQA team clients. When dealing with Scrum-based development teams, the case study SQA team highlighted the following points:

- Developers are very committed to the deadlines set for deliveries to their clients. This eases SQA activity planning, as artifact readiness becomes predictable.
- On the other hand, minor schedule and scope changes are much more frequent, since developers' clients are freer to require changes and these may impact the SQA work. Therefore, SQA should have an agile mindset, too - using Scrum from the SQA side seems a good way to ensure synchronicity between them and development teams.
- Even if products are officially at an initial development stage, Scrum-based teams may adopt other agile techniques that enhance software maturity and anticipate integration and verification efforts. This means that unofficial (i.e. not oriented for obtaining certification credit) SQA-related discussions can take place at early development stages and anticipate potential process nonconformities. Such activities are also important to keep the SQA team up to date with development methods and tools. This technical knowledge improves the effectiveness of an SQA team by enhancing its understanding of software products and strengthening its credibility before development teams. The need for such credibility is stressed by Rierson [Rierson 2013] and plays an important role to empower SQA teams with authority to enforce corrective action.

5.3. Managing SQA activities with Scrum

As stated in Section 4, other works suggest that the peculiarities of SQA activities render agile methods unsuitable for use by SQA teams. However, when these teams are inserted in an agile environment with Scrum-based clients, they must find ways to keep up with the pace of product development.

One simple attempt is to adapt Scrum to the SQA scenario. This approach has a very positive point: SQA sprints can be planned according to the needs of other teams. If a development team sprint features - for instance - an official, "run-for-score" build delivery, the SQA sprint for the same period may feature a build witness activity.

The case study SQA team benefitted from some well-known Scrum positive points: in particular, increased visibility over activity progress and results motivated it to work more efficiently when doing activities that fit in the "short-term delivery" category.

Assessing the efficiency of Scrum is a tricky topic by itself: the pertinence of a given performance criterion is subject to discussion even for more "orthodox" Scrum projects. For example, Santana and Nunes [2014] remark that the Nokia Test (a well-known Scrum efficiency evaluation) should not be used as an absolute measure to compare Scrum efficiency of two or more teams. In our case study, the SQA group struggled to find good metrics for sprint performance evaluation - they also had to be

adapted to SQA reality. In particular, the cross-project nature of our case study team and the long development cycles of airborne software products often meant that it was difficult to assess how team performance evolved over sprints, as the activities changed according to the advancing phases of development projects subject to SQA. Hence, the most meaningful performance indicators in our case study are individual sprint burndown charts - which relate completed activities to planned ones. While the average performance was around 85%, some sprints had lower performance due to sudden priority changes by clients or workload changes caused by impediments (e.g. lack of consensus between teams over a possible process nonconformity).

While common Scrum deliveries include software functionality, SQA teams usually deliver documents: review results, audit reports, nonconformity reports and (besides several other activity result documents) project documentation that shall be maintained by the SQA team, such as the software quality assurance plan. Some of these deliverables are the result of long activities that simply cannot be divided into small ones: it would be difficult to ensure coherency if several people conducted or documented the review of a given artifact. Besides, such reviews may take many cycles of correction and reverification, which makes their execution time quite unpredictable.

In spite of all perceived difficulties, client feedback (in this case, development teams) is positive: the active involvement from the project onset enables the SQA team to have a faster and better understanding of processes and products. Compliance gaps can be identified earlier, thus avoiding potential cost overruns due to late rework.

5.4 Good practices for agile SQA

Some long SQA activities, as described in section 5.3, simply cannot be divided into smaller deliveries. However, it might be possible to perform then within a single sprint window if more team members can work on the same delivery. As the quality assurance team activities are based on sampling criteria - in other words, the SQA choose a predictable amount of life cycle data - its activities can be shared by the team, decreasing the workload of each team member and, consequently, ensuring that long activities can be delivered in time. For example, if a requirement audit needs a certain number of requirement samples but the team judges this number too high to be attained by a single person in a single sprint, two persons or more can perform the audit. Good teamwork is then important to achieve consistent audit results - once more, the Scrum hypothesis of a well-trained, cross-functional team is crucial.

Regarding activities that do not have sampling criteria (write the SQA plan, review plans and standards), it is often difficult to share work among team members. On the other hand, deliveries may be split along several sprints: for instance, one or more document finished chapters, might be considered as a potentially shippable product.

The involvement of SQA teams, at early development stages, on verification activities (requirements, design, coding and integration) in parallel with verification teams can lead to a more proactive thinking instead of a reactive one, anticipating nonconformities and avoiding the hassle of late problem detection and correction.

6. Conclusion and future work

In this paper, we evaluate the effects of Scrum in the workflow of software quality assurance teams in the airborne software domain. The influence of Scrum-based developers and the effects of a Scrum-based SQA process were evaluated by means of a case study: an SQA team presented strengths and challenges it had to face along several sprints and interaction with several Scrum-based development teams. Despite the need for some adaptations, the use of Scrum in SQA also presented advantages and helped the case study team when it needed to keep pace with agile development teams.

Future work includes the evaluation of more advanced management techniques, such as the Scrum of Scrums, to increase synchronicity between development and SQA teams. The good results obtained with Scrum in SQA may open the way for another seemingly unexplored, yet interesting, field of research: the adaptation of practices from other agile methods (such as eXtreme Programming - XP) to the SQA domain.

References

Special Committee 205 (SC-205) of RTCA. (2011) “DO-178C, Software Considerations in Airborne Systems and Equipment Certification”.

Itkonen, J., Rautiainen, K. and Lassenius, C. (2005). Toward an understanding of quality assurance in agile software development. International Journal of Agile Manufacturing, volume 8, number 2, pages 39-49.

Stamelos, I. G. and Sfetsos, P. (2007), Agile Software Development Quality Assurance, IGI Global.

Ambler, S. (2005). Quality in an agile world. Software Quality Professional, 7(4),34-40.

Chisholm, R. A. (2007), "Agile Software Development Methods and DO-178B Certification", Master’s Thesis, Division of Graduate Studies of the Royal Military College of Canada.

VanderLeest, S. H. and Buter, A. (2009) "Escape the waterfall: Agile for aerospace". 28th Digital Avionics Systems Conference, Orlando, USA.

Santana, C. and Nunes A. (2014) "Using Nokia Test to Evaluating Quality and Productivity on Scrum-CMMI Environments". Workshop Brasileiro sobre Métodos Ágeis (WBMA 2014), Florianópolis, Brazil.

Chenu, E. (2009) "Agility and Lean for Avionics". Lean, Agile Approach to High-Integrity Software Conference, Paris, France.

Wils, A., Van Baelen, S., Holvoet, T. and de Vlaminck, K. (2006) "Agility in the Avionics Software World". XP 2006, Oulu, Finland.

Rierson, L. (2013). Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance. CRC Press.

Gomes, André F. (2013). "Agile: Desenvolvimento de software com entregas frequentes e foco no valor de negócio". Casa do Código.

Schwaber, K. and Sutherland, J. (2013). "The Scrum Guide.". Available at <www.scrumguides.org>, accessed in May 2015.