An Assessment of Avionics Software Development Practice: Justifications for an Agile Development Process

Geir K. Hanssen[1(✉)], Gosse Wedzinga[2], and Martijn Stuip[2]

[1] SINTEF, Trondheim, Norway
geir.k.hanssen@sintef.no
[2] NLR, Amsterdam, The Netherlands
{gosse.wedzinga,martijn.stuip}@nlr.nl

Abstract. Avionic systems for communication, navigation, and flight control, and many other functions are complex and crucial components of any modern aircraft. Present day avionic systems are increasingly based on computers and a growing percentage of system complexity can be attributed to software. An error in the software of a safety-critical avionic system could lead to a catastrophic event, such as multiple deaths and loss of the aircraft. To demonstrate compliance with airworthiness requirements, certification agencies accept the use of RTCA document DO-178 for the software development. Avionics software development is typically complex and is traditionally reliant on a strict plan-driven development process, characterized by early fixture of detailed requirements and late production of working software. In this process, requirement changes and solving software errors can lead to much rework, and create a risk of budget and schedule overruns. This raises the question whether avionics software development could benefit from the application of agile approaches. Based on the results of three activities: (1) a literature study on industrial experience with the use of agile methods in a DO-178 context, (2) an expert assessment of the DO-178 objectives, and (3) a survey conducted among European avionics industry, an outline is presented of an agile development process, where Scrum is extended to achieve the DO-178 objectives. The application of agile methods is expected to support frequent delivery of working software and ability to respond to changes, resulting in reduced risk of budget and schedule overruns.

Keywords: Avionics · Certification · Safety critical software · DO-178 · Software Life-Cycle · Agile · Scrum

1 Introduction

Avionic systems play a crucial role aboard modern aircraft. These systems offer pilots operational support in areas such as communications, navigation, and control of the aircraft during all phases of flight and in all weather conditions. A system is safety-critical when its failure could result in loss of life, significant property damage, or damage to the environment [11]. An example of a safety-critical avionic system is the flight control system, which governs the attitude of an aircraft and, as a result, the flight path it follows. Safety-critical systems are not limited to the avionics domain only,

H. Baumeister et al. (Eds.): XP 2017, LNBIP 283, pp. 217–231, 2017.
DOI: 10.1007/978-3-319-57633-6_14

examples of other important domains include, process control [20], medical equipment [17], and automotive [9].

Present day avionic systems are increasingly based on computers and more functions are implemented as software. Certification agencies, like the European Aviation Safety Agency (EASA), accept the use of RTCA document DO-178 [18] for the development of avionics software to provide assurance of compliance with airworthiness requirements. Document DO-178 requires the achievement of many safety objectives, which is generally costly and time consuming [4, 10].

The avionics industry traditionally uses the V-model, or a variant thereof, as life-cycle model for software development. This matches DO-178 well when looking at the life-cycle data items that have to be produced. There are, however, also disadvantages. For example, no working software is produced until late in the development life-cycle. Errors detected in this stage can lead to much rework of earlier performed activities, and increase the risk of budget and schedule overruns [4]. In the same way, changes in requirements in a late stage can also lead to much rework with similar consequences.

The application of agile methods could be a solution for these problems. The difficulty lies, however, in the fact that the looseness of an agile process does not seem to be reconcilable with the rigour imposed by DO-178. For example, agile development considers responding to change more important than following a plan, while DO-178 is strictly plan driven. The main question addressed by this research is how agile methods can be adapted to be usable in an avionics development process that is governed by DO-178.

The following of the paper describes our research method (Sect. 2), an analysis of DO-178C (Sect. 3), an overview of research and industry experience (Sect. 4), a survey of present practice (Sect. 5), and an outline of an agile process aligned with DO-178 (Sect. 6). Conclusions and further work are presented in Sect. 7.

2 Research Method

In order to answer our research question, three complimentary activities have been carried out and used to propose a DO-178-aligned agile process.

(1) An assessment of DO-178 has been performed to indicate how an agile strategy for meeting the objectives could look like and whether there are potential conflicts by using an agile method (Sect. 3.2). Annex A of DO-178 contains 10 summary tables with 71 objectives. The information provided for each objective includes: (a) a brief description, (b) its applicability for each software criticality level, (c) the requirement for independent achievement, and (d) the data items in which the results are collected. Each objective has been assessed to determine how the objective can be met using an agile approach like Scrum and whether there is a need for extensions beyond what can be considered a plain agile approach. The work performed by K. Coetzee[1] was taken as a starting point.

[1] http://www.embeddedfool.net/blog/2015/04/08/a-more-agile-do-178/ (last accessed, Dec. 5, 2016).

(2) Relevant literature addressing the application of agile methods in the avionics domain has been reviewed and main findings about opportunities and limitations of using agile methods for development of avionics software were summarized (Sect. 4). In order to build an understanding of the status of research and reported industrial experience on the use and effects of agile methods in development of safety-critical avionics software, a search for relevant literature has been conducted with Google Scholar. We applied search phrases based on relevant terms such as 'agile', 'avionic', and 'DO-178'. To strengthen the search, we applied snowballing, meaning that relevant work referenced in identified publications was checked for relevance and potentially included if the focus and quality was found sufficient. From this search, 11 publications were found that potentially could offer insight into industrial experience.

(3) A survey was done as an online questionnaire to establish a better overview of the state–including challenges and potential points of improvement–of software development and certification in the avionics industry, and to map the current status of using or plans to use agile methods. As part of the ASHLEY[2] EU-project, we selected professionals believed to have sufficient knowledge about their own organization and about how software is developed and certified. 29 contact persons were selected, each representing a unique ASHLEY partner organization. 10 contact persons completed the questionnaire fully or partially.

Our study has some limitations. Firstly, the literature review identified a relatively low number of relevant studies providing industrial experience. This is however a valuable insight as it nevertheless summarizes the present state of research within this specific domain. Secondly, the survey has a relatively low number of respondents. This is due to resource priorities, but is somewhat compensated by selecting qualified respondents, each representing a major avionic system provider in Europe. The results present the most comprehensive overview of this industry so far.

3 Certification Aspects of Avionics Software Development

3.1 Overview of Document DO-178C

Document DO-178C, "Software considerations in airborne systems and equipment certification" [18] governs the approval of software for avionic systems by certification authorities, such as EASA. In this paper, we simply write DO-178 when referring to revision C of the document.

DO-178 distinguishes five software levels (A–E) based upon the failure condition that may result from erroneous behaviour of the software. Software is classified as (the highest) level A, if erroneous software behaviour can cause or contribute to a catastrophic failure condition of the aircraft, which would result in multiple fatalities, usually

[2] Avionics Systems Hosted on a distributed modular electronics Large scale dEmonstrator for multiple tYpe of aircraft, http://www.ashleyproject.eu (last accessed, Dec. 9 2016).

with loss of aircraft. For lower software levels, the consequence of erroneous software behaviour gradually reduces to no effect on safety (level E).

DO-178 is a process-based standard relying on evidence that the various activities associated with software development have been performed successfully. DO-178 categorizes processes into three types: (1) the software planning process, which defines and coordinates the activities of all processes (2) the software development processes, which produce the software product, and (3) the integral processes, which ensure the correctness of the software product and confidence in the software development processes and their outputs. DO-178 does not address system life-cycle processes, but it does describe the interaction with system processes, including system safety assessment.

Table 1. Assessment of objectives for the software development processes.

DO-178 Objective	Agile Strategy	Remarks
1. High-Level Requirements (HLRs) are developed	A system is divided into features. Features are divided into stories. Stories consist of HLRs (and their test cases)	Features are client-valued functions. At the end of each Sprint, the implemented user stories are used to update the HLRs
2. Derived HLRs are defined and provided to the system processes, including system safety assessment process	Derived HLRs are not directly traceable to system requirements. They are developed in the same way as HLRs (see objective 1)	Derived HLRs are provided to the system processes to determine if there is any impact on the system safety assessment and system requirements
3. Software architecture is developed	Start with a high-level architecture and update/refine it at each software release	Closure activities include a review of the software architecture to make sure it is consistent with the source code
4. Low-Level Requirements (LLRs) are developed	Develop LLRs by defining conditions and associated actions [13]	LLRs can be contained in the source code or the unit tests (embedded in the source code)
5. Derived LLRs are defined and provided to the system processes, including system safety assessment process	Derived LLRs are not directly traceable to HLRs. They are developed in the same way as LLRs (see objective 4)	Derived LLRs are provided to the system processes to determine if there is any impact on the system safety assessment and system requirements
6. Source Code is developed	Develop source code by applying Test-Driven Development (TDD)	Stories are implemented during Sprints
7. Executable Object Code and Parameter Data Item Files, if any, are produced and loaded in the target computer	Develop object code by applying Continuous Integration (CI) and Continuous Delivery (CD)	When a defined set of features is completed, a release will follow

DO-178 provides guidance by (1) stating objectives for software life-cycle processes, (2) describing activities that provide a means for satisfying the objectives, and (3) describing evidence in the form of data items to demonstrate that the objectives have been satisfied. DO-178 does not prescribe a particular software life-cycle or methodology. A software development project defines its software life-cycle by specifying a set of processes and their sequence. The usual sequence through the software development processes is requirements, design, coding, and integration.

3.2 Assessment of Document DO-178C

The assessment revealed that objectives for the software development processes (DO-178, Table A-2) and testing (DO-178, Table A-6) can be achieved by applying agile techniques. The remaining objectives are either outside the agile process or there are no suitable agile techniques to achieve them. These objectives can be achieved using traditional methods (inspections, reviews, analyses, management records).

Table 1 presents the assessment of the 7 objectives for the software development processes (DO-178, Table A-2).

In conclusion, agile methods can be used to achieve a subset of the DO-178 objectives. No prohibitive conflicts have been identified.

4 Overview of Existing Research and Industry Experience

Most of the 11 reviewed publications provide discussions at a conceptual level without any empirical data, indicating that this is a relatively new and immature–but growing–concept within the avionics domain. Some empirical data is presented in only three of the papers. Wils et al. [22] provide some minor insights from the Barco company, Paige et al. [16] present a very small-scale experiment, and Carlson and Turner [1] make a review of five case studies.

This lack of empirical data from industry is in contrast to non-safety-critical domains where the use of agile methods has become common, with correspondingly more empirical research available [6]. One comparable domain, the process control domain, where the IEC 61508 standard applies, is a bit more advanced, but in general it seems that the application of agile methods and techniques to safety-critical software is in its early stages [8]. However, the emergence of literature presenting ideas over the past few years means that the industry is seeking new opportunities for improving their software development processes inspired by other domains.

4.1 Why This Interest in Agile Methods?

The common background and motivation for nearly all reviewed publications is the need for improving the software development process, including certification based on DO-178B/C. The trend seems to be that avionic system complexity is increasing [5]. Requirements tend to be more volatile (even late in the development process), calling for better approaches to manage requirements and their changes in more flexible ways

[5, 15, 16]. We also see an increased customer orientation where industry wants to listen more closely to customers [1, 3, 16, 21, 22], opening up for a more flexible development process with less emphasis on complete and detailed up-front design. Experience also indicates that cost and schedule overruns are happening too frequently [1, 4].

4.2 Evidence and Documentation

Regardless of the process framework, e.g., V-model or an agile process, there is a set of formal data items that has to be produced [5], but an agile process may allow for doing this more efficiently as well as data items may be updated more often. However, if an agile approach is to be used, it calls for some extensions [16], as agile methods, such as Scrum, do not specify such documentation at all. Examples of such data items that are required by certification authorities are the Plan for Software Aspects of Certification (PSAC) and the Software Accomplishment Summary (SAS) [18]. These documents, together with the plans that concern the definition of the life-cycle processes may best be kept outside the agile process.

4.3 More Flexible Management of Requirements and Change

One of the main characteristics of the established practice and application of the V-model is that development of avionics software may be characterized as document driven and sequential [16]. This may become challenging in cases where requirements change throughout a development project, even despite there have been made very detailed plans and design up-front. Change may come from several sources, like design revisions, review of safety analysis, and verification [16]. Recent figures indicate that requirements change can be quite extensive, from 25% in typical projects to 35% in large and complex projects [21], and discovering problems and dealing with changes late in the process may become very costly [4]. According to Wils et al., agile methods may lower the change effort as compared to traditional development [22]. This does not mean that up-front plans are to be avoided, as that would conflict seriously with the process objectives in DO-178. However, the role of agile requirements management is to detail high-level requirements per iteration, not to create new high-level requirements [5]. New high-level requirements could be added after the Sprint, as part of the Sprint review. Up-front requirements may not be complete or even in conflict (and need to be refined) [5].

However, there is a potential conflict here–that flexible requirements management negatively affects the software verification process. If previously verified components of a system are changed, the verification results need to be updated. This requires strict configuration management and relentless testing of the software under development [2].

4.4 Applicability and Obstacles

In general, the consensus seems to be that there is no conflict per se for using agile methods in development of avionics software [2, 3, 13, 21, 22]. In fact XP/Agile is claimed to be particularly suitable [3] to deal with the increasing complexity and

requirements volatility in safety-critical software projects. As changes inevitably do happen, we could make use of better strategies to manage changes.

However, agile methods, such as Scrum, were not designed to support development of large and complex systems like safety-critical avionic systems and there is a lack of techniques and practices to meet the objectives of DO-178. E.g., the requirements for data items and traceability have to be met by setting up a well-functioning framework of tools to support and automate the process to a large extent [3, 22]. An agile process, with short iterations of work, frequent feedback, and evaluation of status and incremental development of the software supports the production of some of the needed data items as part of the development itself. Instead of explicitly producing separate documents, some of the information may be extracted from tools and logs. One of the core objectives of agile methods is to minimize the effort for producing documentation [16, 21]. There is work going on to extend Scrum to make it applicable to regulated domains, for example the SafeScrum framework [14] and R-Scrum [7], which seek to meet requirements mentioned above.

Besides practical aspects of setting up an agile process and a chain of supporting tools, we also need to clarify such a change with the certification authority. A more or less radical change in process will affect the work to be done by this stakeholder and it is of course important that the certification authority representative gets all requested information and eventually gets confidence that the applied approach has led to a safe product without extra problems and in an efficient way.

Besides the core principle of incremental and iterative development, agile methods may also be seen as a collection of practices and techniques. From Chenu [3] and Paige et al. [16], we extracted the following set that may be particularly relevant to safety-critical systems development:

- Test-driven development (need some adaptation, see also [12]).
- Coding standards (already mandatory for DO-178 levels A–C).
- Design improvement/refactoring (creates some challenges with respect to safety analysis [5]).
- The planning game (from XP).
- Emphasis on communication (other than through extensive documentation).

4.5 Team Efficiency and Motivation

One of the main aspects of agile methods is how people work together. As a contrast to plan-based methods where developers take on specialized roles, following detailed plans, agile methods rely on multi-disciplinary teams, with the idea that this better enforces learning and motivation [3]. Furthermore co-located teams are also believed to improve design flexibility and a shared vision of the system under development [1]. A team may also have Designated Engineering Representatives (DERs), who are embedded representatives of the certification authorities within the development team [5].[3]

[3] Under EASA regulation, Certification Verification Engineers (CVEs) perform equivalent tasks as DERs.

4.6 Testing

Extensive testing and full traceability is fundamental in development of avionics software and implementation of all requirements has to be verified by tests [3]. Testing is also strongly emphasized in agile methods, which focus on test-driven development and high test-coverage. However, for avionics software development purposes, agile methods need to extend testing activities–e.g. by having more thorough acceptance testing (not (only) relying on customer feedback) [2, 16]. Carlson and Turner argue that incremental testing increases iteration pace and enables issues to be revealed and dispatched [1]; they also argue that testers should be part of the development team (provided that any independency requirements are guaranteed).

4.7 Adoption of New Software Process Models

Experience (e.g. from object-oriented development) shows that uptake and acceptance of a new practice takes time–we should expect the same for agile methods as well [21, 22]. The avionics domain relies on well-established and well-proven practices and processes and it is natural to be careful with new ideas, like agile methods, as they may seem to impose more challenges than benefits. However, as this literature in sum shows, there seems to be a growing interest at least.

4.8 Relating Findings to Other Domains

The literature review done here has focused explicitly on the avionics domain. However, we find that the main challenges and approaches clearly coincide with other domains where safety-critical software is essential. Other studies show that the same type of challenges are being addressed, e.g., for process control systems [20], medical equipment [17], and automotive [9], and that agile methods may be applicable to other safety standards and frameworks like IEC 61508, SPICE, and IEC 62304.

5 Survey to Assess Present Practice

A questionnaire was used to gain insights into the organizations' profiles, their maturity, their relationship to safety standards and authorities, various life-cycle aspects, and perceived challenges and problems.

5.1 Respondents' and Organizations' Profiles

Respondents have a great variety in profiles, from developers and testers to managers. Their organizations also have a wide range of business models, target markets (civil passenger aircrafts on the top), and type of software applications (real-time embedded systems being the most common).

5.2 Maturity

The avionics domain/industry is mature and professional with established system providers having decades of experience. There is a wide range of methods for requirements analysis and architectural and detailed design in use. There is also a wide range of testing approaches in use (white/black-box–unit/module/system/hardware-in-the-loop). All practice extensive testing and inspection. Customer involvement is extensive. There is extensive use of DOORS® from IBM Rational for requirements analysis and management, but half of the respondents also use typical office tools.

5.3 Relationship to Safety Standards and Authorities

DO-178 is clearly the most relevant standard for all organizations. Applications are developed at all levels of DO-178, where level C is the most common (60% of the respondents). Consequently, there is a very high coverage of data items. When asked about the level of interaction with the external assessor, 50% report that they collaborate with the assessor in all phases of the project. The rest report a lower level. The average estimate of costs related to verification and certification (including all reviews and testing) is 40% of the total project budget.

5.4 Life-Cycle Aspects

There are a wide variety of software life-cycle models in use. The V-model is in use in some form by all organizations, while 25% use incremental/iterative methods in some form. Customers are involved to a very high degree. Testing (in general) and code inspection/analysis are used by all respondents. Formal methods are applied by about a third of the respondents.

5.5 Perceived Challenges and Problems

The top challenges with respect to verification and certification include: (1) having sufficient resources, infrastructure, and competency/staff, (2) having sufficient quality of customer communication, including requirements specification and feedback, and (3) demonstrating compliance with DO-178 requirements to certification authority. The top-rated problems with the software development process are requirements management (frequent changes, insufficient requirements, ambiguous requirements, and addition of new requirements), late discovery of problems/defects, and project cost overruns.

6 Towards an DO-178-Aligned Agile Approach

As mentioned in Sect. 3.1, document DO-178 [18] does not prescribe a particular software life-cycle model. This makes it possible to define software life-cycles, such as, waterfall, V-model, incremental, and spiral, but also to apply agile methods. Scrum is considered to be a suitable (non-safety) agile framework that could be used as a baseline.

It is the most commonly used agile framework in the software industry, in general, with a large number of training resources, industrial experience, and available research literature. Scrum will have to be extended for the development of avionics software to enable delivery of all required data items in compliance with DO-178.

6.1 Scrum Phases

In his seminal paper [19] on the Scrum development process, K. Schwaber made a distinction into the phases Pregame, Game, and Postgame. In this paper, we use the terms Preparation, Development, and Closure, which are also frequently used, e.g., [13]. Applying the Scrum phases to the software development and software verification processes of DO-178, as depicted in Fig. 1, allows the mapping of agile methods to these processes.[4]

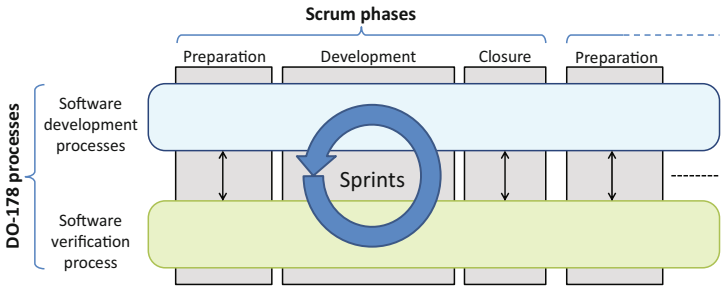

Fig. 1. Application of Scrum phases to DO-178 processes.

During the Preparation phase, planning and architecture activities are performed. Scrum's concept of planning is somewhat broader than that of DO-178. Scrum includes the definition of the next software release based on the currently known backlog, analysis of system requirements, and development of user stories. The architecture activities establish (or update) the software structure. During the Development phase, the functionality of a new release is developed as well as tests for new or changed code. The software is designed, and source code is implemented, integrated, and tested during a sequence of Sprints. In the Closure phase, the software release is prepared, including system testing, final documentation, and release. The sequence of Preparation, Development, and Closure is repeated until the final software release has been completed. In the next sections, the activities in each phase are described in more detail.

6.2 Preparation Phase Activities

During the Preparation phase, the allocated system requirements, or a subset thereof, are taken and high-level requirements (HLRs) are produced in the form of features that

[4] For simplicity, the DO-178 planning process and integral processes other than software verification are not shown in Fig. 1.

are further divided into user stories. A software architecture is established (or refined), which, together with the prioritized HLRs, as part of the product backlog, is provided to the Development phase. As required by DO-178, outputs of all processes are verified, e.g., by means of review or analysis. Further details are presented in Table 2.

Table 2. Activities during Preparation phase.

DO-178 Process	Inputs	Activities	**Outputs**
Software requirements	Allocated system requirements, software level	Define system features and prepare user stories. A story consists of HLRs	HLRs, trace data
Software design	HLRs	Establish or refine software architecture, including partitioning concept	Software architecture, trace data
Software verification	HLRs, software architecture, trace data	Define test cases for HLRs. Verify all outputs	HLR test cases, verification results

The planning process of DO-178 is kept outside the agile process. It is responsible for establishing and updating all plans, including the Software Development Plan, the Configuration Management Plan, and the Plan for Software Aspects of Certification. The latter document is used for communication with the authorities.

6.3 Development Phase Activities

The Development phase consists of a sequence of Sprints, all with preferably the same fixed duration (from 1 to 4 weeks). The number of Sprints is not fixed. The result of a Sprint is a set of implemented and tested user stories that are integrated into a working application. In addition, a Sprint produces information for the assessor (the data items). The application can be demonstrated to stakeholders, but not all features may be complete and hence it is not releasable. Further details are presented in Table 3.

Agile development promotes the Test-Driven Development (TDD) technique. A cyclic process is performed whereby first LLRs are established together with their test cases. Next, test code is produced and all tests are executed to verify that they fail. Then, source code is produced that just passes the tests. Finally, the code is refactored and tests are re-executed. This cycle repeats until all LLRs have been implemented. In practise, the TDD technique implies that software development activities will be performed in conjunction with software verification activities.

6.4 Closure Phase Activities

Upon start of the Closure phase, a sufficient number of features should be completed to warrant release of the application. During Closure, all data items that already exist in some form (see outputs in Tables 2 and 3) are brought up to date. The remaining data

Table 3. Activities during Development phase.

DO-178 Process	Inputs	Activities	**Outputs**
Software design	HLRs, software architecture, trace data	Define Low-level requirements (LLRs) by conditions and associated actions [13]	LLRs, trace data
Software coding	LLRs	Produce code for the LLRs	Source code
Integration	Source code	Perform continuous integration	Executable object code
Software verification	HLRs, HLR test cases, software architecture, LLRs, source code, executable object code, trace data	Establish test cases for LLRs. Produce test code for HLRs and LLRs. Execute (automated) tests. Verify all outputs	HLR test procedures, HLR test results, LLR test cases, LLR test procedures, LLR test results, verification results

items required for compliance with DO-178 are produced by other processes than software development and software verification. For example, the software configuration process produces the Software Configuration Index and the certification liaison process produces the Software Accomplishment Summary.

6.5 Remarks and Potential Issues

The proposed process aims to address some of the key challenges we identified in the survey, in particular challenges related to requirements management. Breaking work down in shorter iterations, including planning (Preparation) and evaluation (Closure) means that planning may be done using updated information from previous Sprints, and that each Sprint provides information needed to meet the requirements of DO-178 (in the form of data items). From related research we know that such a process needs to be supported by tools to automate test-driven development and documentation creation as much as possible in order to save time and to ensure quality and consistency [8].

Including agile approaches in the development process for avionics software promises the usually cited benefits such as frequent delivery of working software, including all data items required by DO-178, and the ability to deal with frequent changes in requirements. There are, however, also a number of potential issues.

Contrary to the waterfall model, or the V-model, HLRs are defined in batches; each time that the Preparation phase is entered, a sufficient number of HLRs are defined for the subsequent sequence of Sprints. Having no overview of the complete set of HLRs in an early phase of the development could lead to an inadequate software architecture that may need drastic (and therefore costly) revision during subsequent Preparation phases. This means that also agile projects needs to invest in a sufficient level of detail of HLRs and overall system architecture early. An agile process though may create better opportunities to manage changes when they occur.

Another issue is that the definition of derived HLRs late in the development, e.g., after several cycles of Preparation, Development, and Closure have taken place, may have consequences for the safety analysis [21]. For example, if derived HLRs imply new interfaces that falsify earlier independence claims, a higher software level could be required, creating additional (verification) work that could have been done more efficiently when known beforehand.

7 Conclusions and Further Work

The development of safety-critical software by the avionics industry is governed by RTCA document DO-178. The document places much emphasis on documented and traceable verification to achieve an acceptable level of confidence that the software development activities have been performed successfully. Indeed, our survey, among major players in the European avionics industry, confirmed that verification and certification constitutes a large portion of the total costs of development (estimated 40%). The survey also revealed other challenges perceived by this industry, including requirements volatility, late discovery of problems/defects, and project cost overruns.

The adoption of an agile framework could be a solution for these challenges; this is in line with other related safety-industry oriented research [7, 8]. At present, the lifecycle model mostly used by the avionics industry to organize software development is the V-model, or variants thereof. DO-178, however, does not preclude the use of any particular model, and in general, there seem to be no obstacles for adopting an agile framework. It is clear that agile methods, like Scrum, need to be adapted to fit in the development and certification of avionics software. In particular, such methods need to be extended to fulfil requirements of traceability and documentation. Some of these may be enabled by use of proper tools that provide a high level of automation.

Using Scrum as a basis, an approach has been outlined that benefits from agile methods and can also satisfy the objectives of DO-178. Some DO-178 objectives are achieved in an agile way, while others, in particular a subset of the verification objectives, are achieved by traditional means (management plans, reviews, and analyses). Benefits expected from the agile approach include reduction of risks, adaptability to changing requirements, and overall a reduction of development cost.

There are, however, issues that need further investigation. One of these is that software requirements are defined in batches; each time, sufficient software requirements are defined for the subsequent sequence of Sprints. Having no overview of the complete set of software requirements in an early phase of the development could lead to an inadequate software architecture that would need thorough revision later on.

To conclude, agile methods may promise to resolve some of the specific challenges in the avionics domain, but there is still a clear need for more research and industrial experimentation to verify applicability and to demonstrate improvement effects.

Acknowledgments. The authors would like to thank the anonymous contributors to the survey and Rob Udo from NLR for his contributions to this research. Also the insightful comments from the reviewers are much appreciated. The research leading to these results has received funding

from the European Community's Seventh Framework Programme FP7/2012-2016 under grant agreement no. ACP2-GA-2013-605442.

References

1. Carlson, R., Turner, R.: Review of agile case studies for applicability to aircraft systems integration. Procedia Comput. Sci. **16**, 469–474 (2013)
2. Cawley, O., Wang, X., Richardson, I.: Lean/Agile software development methodologies in regulated environments – state of the art. In: Abrahamsson, P., Oza, N. (eds.) LESS 2010. LNBIP, vol. 65, pp. 31–36. Springer, Heidelberg (2010). doi:10.1007/978-3-642-16416-3_4
3. Chenu, E.: Agility and lean for avionics. In: Lean, Agile Approach to High-Integrity Software Conference, Paris (2009)
4. Chenu, E.: Agile and Lean software development for avionic software. Whitepaper, Thales Avionics (2011)
5. Coe, D.J., Kulick, J.H.: A model-based agile process for DO-178C certification. In: Proceedings of 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing, Las Vegas (2013)
6. Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.B.: A decade of agile methodologies: towards explaining agile software development. J. Syst. Softw. **85**(6), 1213–1221 (2012)
7. Fitzgerald, B., Stol, K.-J., O'Sullivan, R., O'Brien, D.: Scaling agile methods to regulated environments: an industry case study. In: Proceedings of the 2013 International Conference on Software Engineering. IEEE Press (2013)
8. Hanssen, Geir K., Haugset, B., Stålhane, T., Myklebust, T., Kulbrandstad, I.: Quality assurance in scrum applied to safety critical software. In: Sharp, H., Hall, T. (eds.) XP 2016. LNBIP, vol. 251, pp. 92–103. Springer, Cham (2016). doi:10.1007/978-3-319-33515-5_8
9. Hantke, D.: An approach for combining spice and scrum in software development projects. In: Rout, T., O'Connor, R.V., Dorling, A. (eds.) SPICE 2015. CCIS, vol. 526, pp. 233–238. Springer, Cham (2015). doi:10.1007/978-3-319-19860-6_18
10. Hilderman, V.: DO-178B Costs Versus Benefits. HighRely Inc., HighRely Whitepaper (2009)
11. Knight, J.C.: Safety critical systems: challenges and directions. In: Proceedings of the 24rd International Conference on Software Engineering, ICSE 2002. IEEE (2002)
12. Lambourg, J., Comar, C.: Methodology: agile development of safety critical systems. OpenCoss Framework 7 project (2012)
13. Meunier, V., Destouesse, M., Cros, T.: How to "take credit" of agile principles within a certification context? (2008) (Presentation)
14. Myklebust, T., Stålhane, T., Hanssen, G., Wien, T., Haugset, B.: Scrum, documentation and the IEC 61508-3: 2010 software standard. In: International Conference on Probabilistic Safety Assesment and Management (PSAM). PSAM, Hawaii (2014)
15. Paige, Richard F., Charalambous, R., Ge, X., Brooke, Phillip J.: Towards agile engineering of high-integrity systems. In: Harrison, Michael D., Sujan, M.-A. (eds.) SAFECOMP 2008. LNCS, vol. 5219, pp. 30–43. Springer, Heidelberg (2008). doi:10.1007/978-3-540-87698-4_6
16. Paige, R.F., Galloway, A., Charalambous, R., Ge, X.: High-integrity agile processes for the development of safety critical software. Int. J. Crit. Comput.-Based Syst. **2**(2), 181–216 (2011)
17. Rottier, P.A., Rodrigues, V: Agile development in a medical device company. In: AGILE 2008 Conference (2008)
18. RTCA, DO-178C: Software considerations in airborne systems and equipment certification (2011)

19. Schwaber K.: SCRUM development process. In: Sutherland, J., Casanave, C., Miller, J., Patel, P., Hollowell, G. (eds.) Business Object Design and Implementation, pp. 117–134. Springer, London (1997). ISBN 978-3-540-76096-2
20. Stålhane, T., Myklebust, T., Hanssen, G.K.: The application of Scrum IEC 61508 certifiable software. In Proceedings of ESREL, Helsinki, Finland
21. VanderLeest, S.H., Buter, A.: Escape the waterfall: agile for aerospace. In: Proceedings of IEEE/AIAA 28th Digital Avionics Systems Conference, DASC 2009, p. 6, (6D3). IEEE (2009). doi:10.1109/DASC.2009.5347438
22. Wils, A., Baelen, S., Holvoet, T., Vlaminck, K.: Agility in the avionics software world. In: Abrahamsson, P., Marchesi, M., Succi, G. (eds.) XP 2006. LNCS, vol. 4044, pp. 123–132. Springer, Heidelberg (2006). doi:10.1007/11774129_13