sciendo

TRANSACTIONS ON AEROSPACE RESEARCH
2(259) 2020, pp.59-71
DOI: 10.2478/tar-2020-0011
eISSN 2545-2835

SAFETY CRITICAL SOFTWARE DEVELOPMENT METHODOLOGIES IN AVIONICS

Pawel Zakrzewski[1,2] • ORCID: 0000-0001-8302-6078
Janusz Narkiewicz[2] • ORCID: 0000-0002-4701-3285
Darren Brenchley[3] • ORCID: 0000-0002-5191-1184
[1] GE Company Polska Sp z o.o., Aleja Krakowska 110/114, 02-256 Warsaw, Poland
[2] Warsaw University of Technology, Plac Politechniki 1, 00-661 Warsaw, Poland
[3] GE Aviation Systems Ltd, Cheltenham, Gloucestershire GL52 8SF, UK. Incorporated in England No. 745917
pawel.zakrzewski@ge.com, janusz.narkiewicz@pw.edu.pl, darren.brenchley@ge.com

Abstract

This article summarizes avionics safety-critical software development methodologies and implications of the DO-178C standard from an Agile application perspective. We explain the safety-critical software categorization. It also outlines the main differences and advantages of different approaches to the development process, from Waterfall through the V-model to Iterative and Incremental. Agile principles are explained as well as a Scrum − which is a popular framework in the non-safety-critical software industry. The application of Agile, for safety-critical software considerations, is based on the practical knowledge of the authors, and looks at the potential solution from a DO-178C standard, size of the project, scalability, and organizational culture points of view. Definition of the Agile type of framework, consistent with the certification process and existing standards, has been highlighted as a potential game-changer for the avionics industry.

Keywords: Avionics, Safety Critical Software, Agile, Systems Engineering.

SOFTWARE IN AVIONICS SYSTEMS

Almost all avionics software is qualified as safety-critical, so the development process is strictly regulated and often requires certification. Most of the software is developed according to the DO-178C standard. This standard defines the goals to be achieved and the requirements that should be met, as a necessary condition for the final system certification. However, it doesn't define the process of the software development.

DO-178C requires the creation of detailed documentation and fully traceable requirements. Traditional interpretations of this and other standards, drives aerospace companies towards application of the waterfall method for the management of avionics projects, which is consistent with the project management methodology for the whole platform − aircraft. The waterfall approach involves decomposition of the detailed project plan, into connected phases which are chronologically accomplished.

Currently Agile methodology, expressed by a variety of frameworks, is widely used for non-safety-critical software product development, and is regarded as more effective than a traditional waterfall approach. However, in reality, Agile frameworks have limited usage in the development of safety-critical aviation software, due to the certification process constraints. Agile frameworks do not properly address the requirements of traceability, as the product in progress is a subject of constant changes, architecture is constantly modified in iterative and incremental processes, which is accompanied by the refactoring of the code. Considering this highly variable development process, it is very difficult to assure traceability in the product development. An Agile methodology does not cover the creation of documentation as an essential part of the project, however it is a necessary part of the avionics product certification.

Another potential problem not addressed by Agile methodology, is recognition and inclusion of certification authorities as a first-class project stakeholder. Without their involvement on a continuous basis, the potential benefits of Agile methods are difficult to realize. The implication of these limitations is that Agile is not yet widely applied for the development of safety-critical avionics products.

The creation and implementation of an Agile framework that addresses all safety-critical software limitations, and which fully supports the DO-178C standard would be a turning point for the industry. In light of successful implementation in other industries, Agile methods would potentially allow for a reduction in both time and cost of product development, which are key constraints on the introduction of new technologies in avionics.

SAFETY CIRITICAL SYSTEMS AND DO-178C STANDARD

A safety-critical system is a system whose failure or malfunction may result in one (or more) of the following outcomes: death or serious injury to people, loss or severe damage to equipment or environmental harm. [1]

The DO-178C, Software Considerations in Airborne Systems and Equipment Certification document is the primary standard used by certification authorities such as FAA and EASA, to approve software-based aerospace systems for commercial use. The document is published by Radio Technical Commission for Aeronautics (RTCA) and provides guidance for determining, in a consistent manner and with an acceptable level of confidence, that the software aspects of airborne systems and equipment comply with airworthiness requirements [2]. The latest issue is called DO-178C and was approved by the RTCA in December 2011.

The standard does not define the process of the software development itself, but objectives to be met. The DO-178C requires the creation of detailed documentation and assurance of full requirements traceability.

The Software Design Assurance Level (DAL) categorizes software by potential consequences of failure and needs to be established for certification purposes. It is determined based on the safety assessment process and hazard analysis, by analysing the effects of a failure condition in the developed aerospace system. The failure conditions are categorized by their effects on the aircraft, crew, and passengers [3]. Software intended to control the aircraft and monitor safety-critical systems receives the highest DAL A (for example the aircraft control system or fuel monitoring system).

The DAL categories are as follows:

Level A − Catastrophic − Failure may cause deaths, usually with loss of the airplane.

Level B − Hazardous − Failure has a large negative impact on aircraft safety or performance; it may reduce the ability of the crew to operate the aircraft due to physical distress or a higher workload or causes serious or fatal injuries among the passengers.

Level C − Major − Failure significantly reduces the safety margin or significantly increases crew workload. It may result in passenger discomfort (or even minor injuries).

Level D − Minor − Failure slightly reduces the safety margin or slightly increases crew workload. Examples might include causing passenger inconvenience or a routine flight plan change.
Level E − No Effect − Failure has no impact on safety, aircraft operation, or crew workload.

Certification authorities require the correct DAL to be established by the applicant during the 'effects of potential failure' analysis and assessment process. To comply with DO-178C, evidence of meeting safety objectives needs to be demonstrated. The necessary rigor of evidencing the safety objectives depends on the DAL.

The traceability of the certification artefacts is a key requirement of DO-178. A documented relation between every line of source code in the software tracing up to Low Level Requirement (LLR) then to High Level Requirement (HLR). Traceability allows us to check whether each requirement is satisfied by the software and tested accordingly. It also ensures, that the system is complete from the requirements point of view and every line of the software code has a purpose.

METHODOLOGIES OF DEVELOPMENT

Waterfall process

The Waterfall methodology is a sequential project management approach, where stakeholder and customer requirements are gathered at the beginning of the project, and then a sequential project plan is created to accommodate those requirements [4]. The waterfall approach is the most popular project management approach across various industries and research organisations. Chronological sequencing makes it easy for implementation and understanding in all engineering domains, from construction to IT and software development.

In the waterfall process, the project is organised within five to seven phases following in a sequential order. To strictly follow the model, usually every phase should be accomplished before the next one can be started.

The Waterfall process phases according to the Royce [5] definition are requirements formulation, design, implementation, verification and maintenance (figure 1):

Requirements: The main assumption in waterfall is that all stakeholder/customer requirements are fixed, known and collated into one consistent document at the beginning of the project. It allows us to build a detailed plan for each project phase. This approach assumes no further stakeholder/customer involvement until the product is completed.

Design: The design phase starts from the requirement analysis and is followed by defining theoretical solutions. The next part of this phase is translation of these ideas into a feasible solution ready for implementation.

Implementation: The implementation phase involves writing the code which expresses the selected solution and corresponds with the requirements.

Verification: This phase checks the product developed meets all the requirements defined at the beginning of the project. The completed product is released to the customer.

Maintenance: When the product is in regular use, errors or imperfections may be identified that require repair or enhancement. The development team implement these corrections and improvements to sustain product applicability and usage.

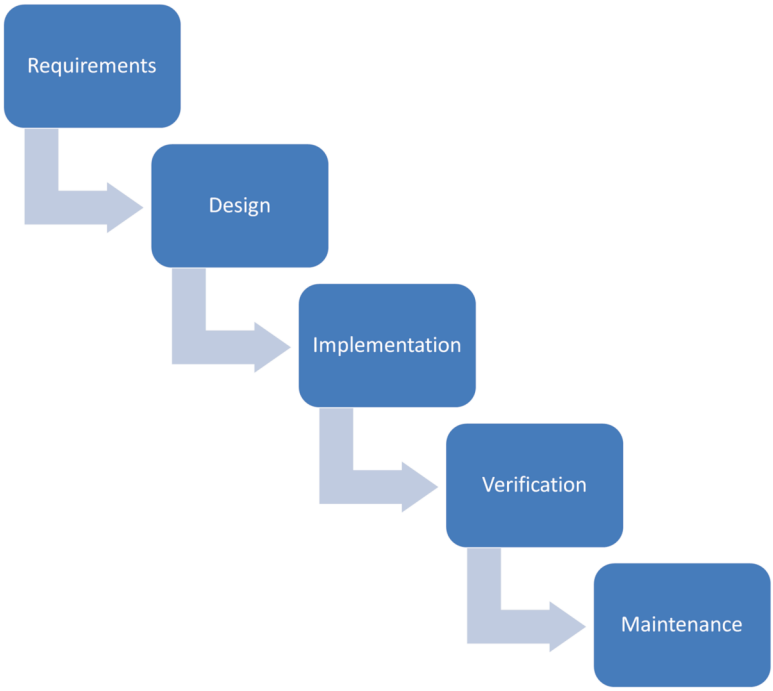

Fig. 1. Waterfall process.

If the project goes according to plan these phases of the Waterfall model are followed consecutively, with no reiteration. Repetition of phases is required only if, during the verification or maintenance phase, significant problems or failures are identified.

There are many constraints concerning the application of Waterfall-based engineering product development. Largely these issues are due to the "rigidity" of the process, the sequential development, lack of customer involvement in the product development process after the requirements phase, and the limited feedback to successive steps of the process.

Initial phases of projects often suffer from inadequately defined customer needs, which can lead to "unstable" requirements at this stage, subject to change as the needs are clarified. Having architecture design based on these unstable requirements, exposes the project to the risk of substantial reworking due to the necessary implementation of fundamental changes to meet customer requirements. Having some prototypes or initial design demonstrated to the customer would be a good solution for validating customer needs, but in Waterfall this is only available very late in the process. The implementation of changes late in the product lifecycle (for instance, following stakeholder feedback on a prototype, or as a result of integration or verification activities) is both challenging and costly to implement, and becomes more difficult as the project progresses. The high risk and uncertainty generated is detrimental to the success of the project [6].

The Waterfall process also has advantages. The approach is intuitive, logical and chronological, which makes the project easy to plan. It also helps both the customer and the project team to understand how the project will be executed. As an outcome of the planning phase, estimated dates and activities are sequenced in a chronological order. This is often expressed in a Gantt chart.

An essential part of the Waterfall process is the initial understanding and completeness of the requirements leading to an established design definition, prior to commencement of the implementation phase.

This stringent approach ensures the integrity of systems engineering practices and makes the monitoring and control of a specified project schedule more straightforward. It helps to avoid problems with traceability and makes implementing selected functionality simpler as the systems definition evolves during project execution.

The V-Model of product development

The V-Model is a visual representation of a Systems/Software Development Life Cycle (SDLC), (Figure 2). Starting with customer needs in the top left-hand corner of the diagram, each step down the left of the V-Model provides further levels of definition and decomposition [15]. In the steps defined as Systems Requirements, High-Level Design and Detailed Design we reach the Implementation phase at the bottom of the diagram. During this phase the specialists from Software and Hardware receive specific requirements and a detailed architecture which provides the necessary framework to write software code and to implement it within dedicated hardware [15].

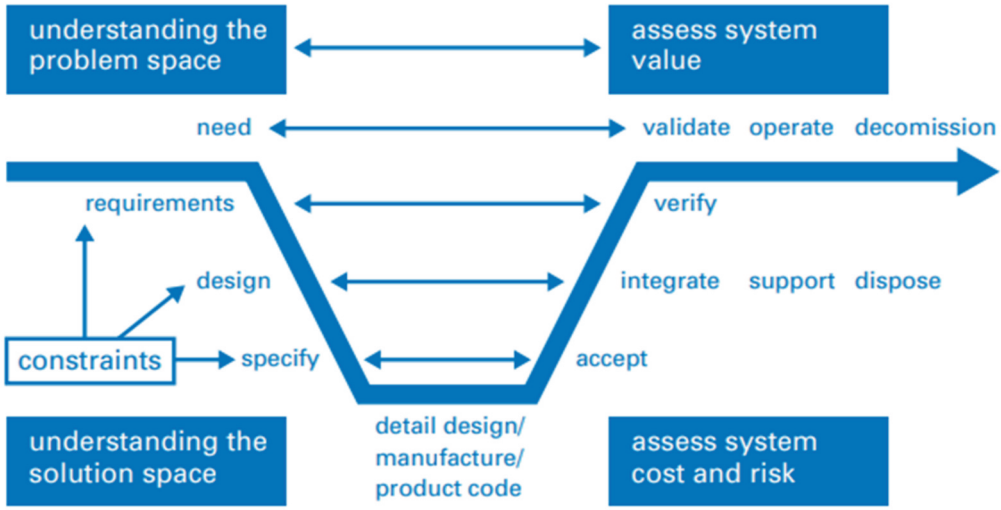

Fig. 2. Vee/V-Model.

Unlike the Waterfall-Model, which cascades down in a sequential process, the V-Model pivots upwards from the implementation phase to form the vee. This demonstrates the interdependence of development and testing activities. The testing can be categorised as follows:

Verification: "The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Contrast with validation" i.e. has the product been developed correctly [9,10,13]
Validation: *"The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders."* i.e. has the right product been developed [9,10,13]

For this reason, the V-Model is also known as the Verification and Validation Model [7]. This methodology of linking both development and testing activities is essential in high integrity environments and this explains its popularity within various sectors of technologies, where traceability is crucial to success.

A common misunderstanding in the use of the V-Model is to consider it as an extension of the Waterfall Model. Sometimes project teams run through each phase sequentially down the left-hand side and then back up the right, without appreciating the horizontal linkage between the development phases and the testing. However, Test Engineers must be involved in reviewing development information as soon as possible, in order to appreciate the true value of this Verification and Validation Model. This often results in a more labour-intensive development process which could indicate that this model is not suitable for projects where several changes are likely to occur [7].

Just like the Waterfall Model, the cost and time of finding and fixing unplanned changes greatly increases as the project progresses. The V-Model's strength is its adherence to documentation, but this can also be its disadvantage. If any requirement changes it may impact other requirements, which can cause significant rework of documentation. Also, the separation into horizontal bands of the Validation and Verification can lead to situations when the project team, having undertaken the requirements elicitation with the customer, then work for a long period of time following the 'internal' elements of the V-Model without communicating with the customer. This model does not assure a culture of interaction with the customer other than at the beginning and the end of the process. This means inherent resistance to customers changing needs and for this reason, the V- Model is often considered to be rigid and inflexible.

It may be also argued that the V-Model has been the domain of the Systems/Software development life cycle. Its adherence to what is, in effect, the project development life cycle excludes the traceability of requirements that precede contract award, and which were gathered during the procurement phase [16], [8]. In a long cycle business like Aviation this could be considered a drawback, especially due to the increased risk of changes in stakeholders needs that can likely occur over extended timeframes.

The V-Model, although often misused, is a very common model and forms a central tenet of the ARP4754A Guidelines for Development of Civil Aircraft and Systems [12]. This set of guidelines highlights the value of the V-Model in that it can be applied at multiple levels to represent the wider system of interest. At the Aircraft platform the high-level requirements and architecture become the concept of operation (customer needs) for the Tier One Suppliers, and these high-level requirements in turn become the customer needs for the next tier down and so on throughout the whole supply-chain.

The other advantage of the V-model is the consideration given to verification whether the product is being built correctly, and information that the right product was created for the customer occurs much earlier in the project life cycle. For each phase of development there is a corresponding test plan. Therefore, if the model is followed correctly, the documentation created will satisfy the necessary requirements traceability that certification bodies like EASA and the FSA mandate.

Iterative and incremental development

The background of Iterative and incremental product development is that the system is considered as a composition of parts, and each part is then subjected to a full development cycle – requirements, design, build, test [6]. With every iteration, the system develops incrementally, growing towards the final product. Each iteration creates the opportunity for design modifications and adding new functionalities and should be considered as a waterfall process or V-model within the scope of each iteration. It is important and useful to define a system as a composition of functionality blocks. A development team can work on one or more blocks concurrently and run full lifecycle iterations on them. With every iteration, a product developed approaches the final product stage by planned increments. When all the blocks have been built, the development team performs validation and verification activities on the whole system.

The V-Model can also be used in more advanced ways to enhance an Incremental and Iterative approach. For example, with Incremental the requirements and architecture are captured once on the left-hand side of the V-Model and feed multiple subsystem implementation phases. It has the advantage of leading to potentially earlier releases and easier testing. However, issues within the system requirements and design processes may be discovered late, for instance during the implementation phases, which could cause costly changes to multiple increments of the subsystems [11] (Figure 3).

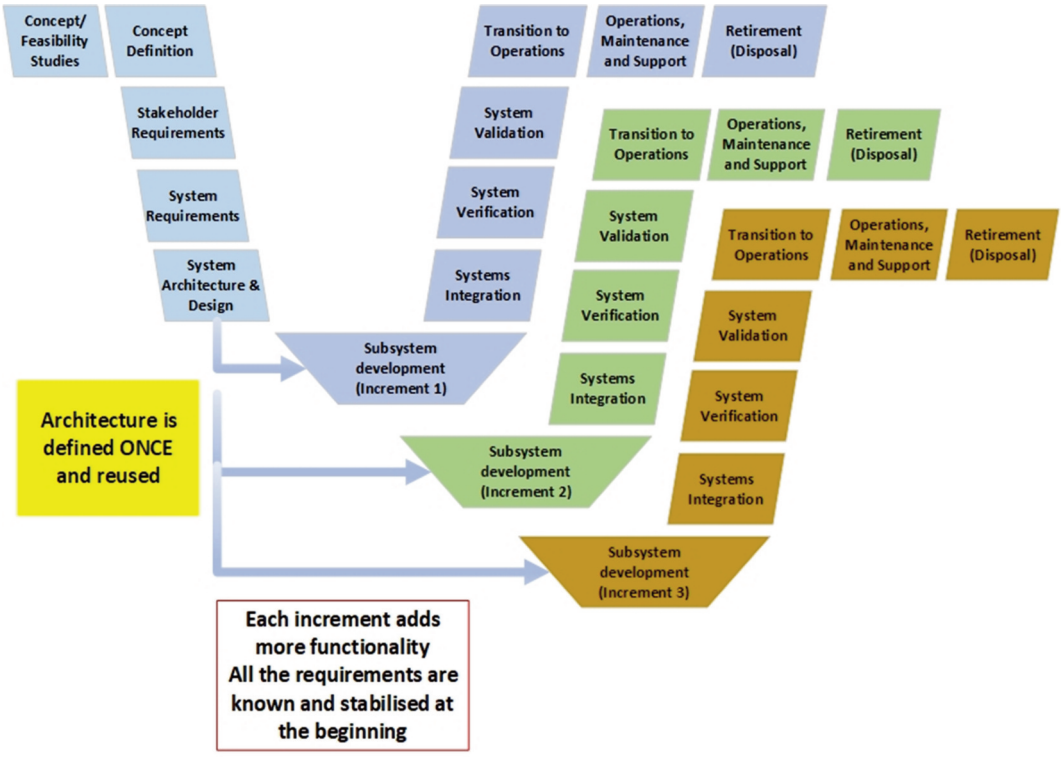

Fig. 3. Incremental V-Model (Presland, 2019).

The V-Model may also be used in an evolutionary / iterative way. The main difference from the incremental approach is iterating through the both sides of the V-model instead of just the right verification and validation side focus. The Requirements and Architecture can be refined and changed within each evolution, based on the lessons learnt during the previous iteration. This has the advantage of involving the customer at multiple points during the project life cycle and especially where there is ambiguity in the initial requirements [11], (Figure 4). However, it is likely that it would be difficult to estimate how many increments might be required to complete the project.

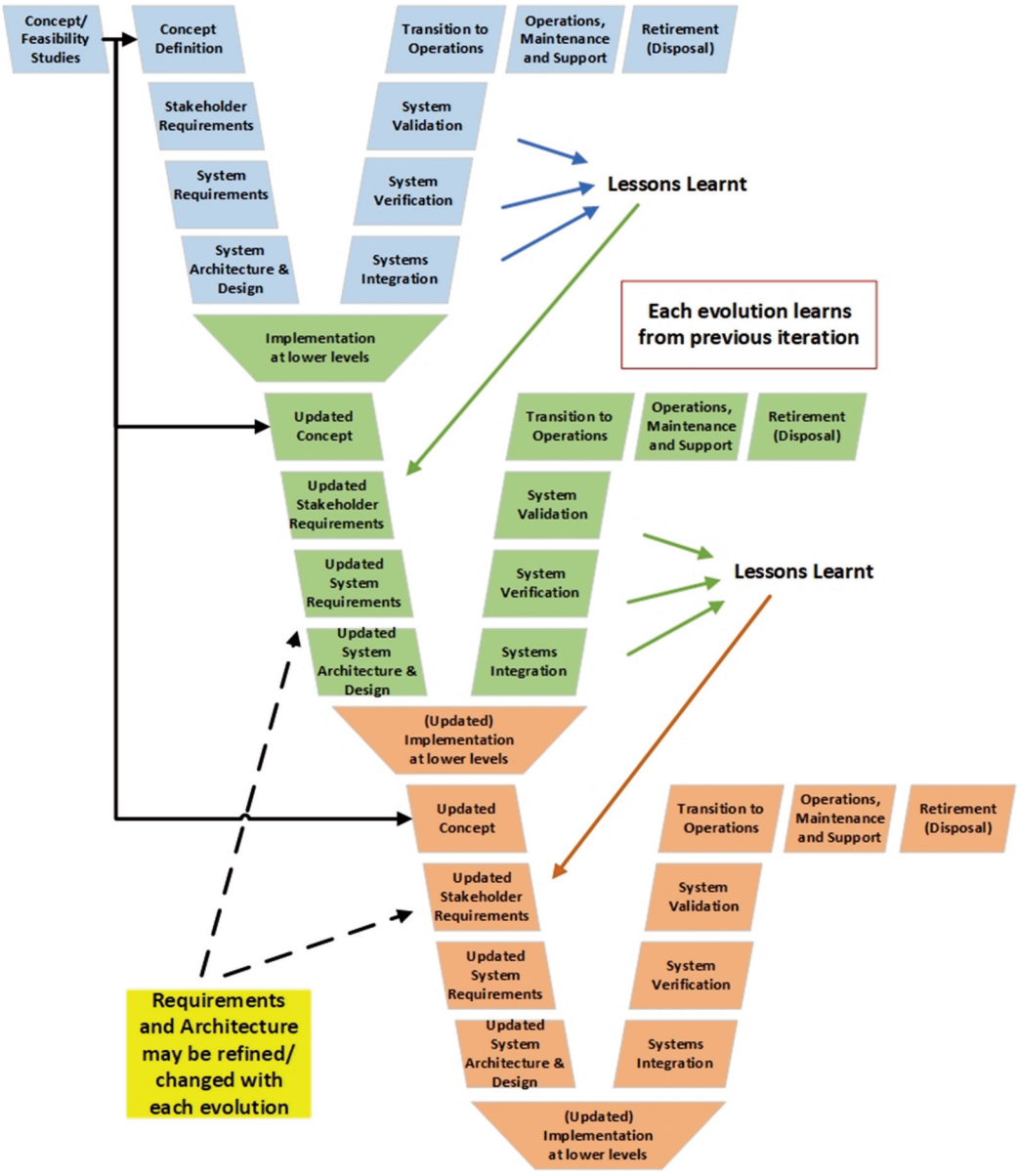

Fig. 4. Evolutionary (Iterative) V-Model (Presland, 2019).

Agile process

The Agile process is fundamentally different from the Waterfall in two main aspects. It is not a chronological sequence of stages, and it assumes by default customer involvement throughout the product development cycle. The Agile method is the iterative process of delivering the product in stages to allow collection of customer feedback that will be implemented later. It does not require deep customer involvement to establish the requirements at the first stage of the process.

The „Manifesto for Agile Software Development" [18] is a document published in 2001 by the creators of this methodology, where they defined the main values and principles. The Agile Manifesto contains four fundamentals of Agile.

The Agile Manifesto states:

- Individuals and interactions over processes and tools,
- Working software over comprehensive documentation:
  - Customer collaboration over contract negotiation,
  - Responding to change over following a plan.

It means that competent people who work effectively as a team are more important, from the product point of view, than process followers. It also emphasises that the ultimate end goal of the project is to have an adequately developed product, rather than perfect documentation. The Agile Manifesto encourages collaboration with the customer and stakeholders, to understand their changing needs and collect feedback during product development as a key priority. The last point of the Manifesto opens up the method for smooth and easy change implementation rather than rigid, long term and detailed planning.

Scrum framework

The Scrum is the most popular framework adopted by industry, and is an essential representation of Agile fundamentals. It defines components of the process, i.e. teams (with roles assigned to individuals), events, artefacts and rules. Every component of the framework has a defined role and should be executed properly to successfully develop the product in a consistent way. The development work is broken down into Sprints i.e. the goals that need to be achieved during each iteration, in timeboxed periods no longer than one month (figure 5). Progress is tracked daily, in the form of short meetings focused on defining daily goals and removing potential obstacles. Every Sprint is summarized by the Sprint review where the accomplished work is demonstrated.

The Agile methodologies, expressed by a variety of frameworks, are widely used for non-safety-critical software development and appear to be more effective than the traditional waterfall method, especially when addressing possible requirement modifications.

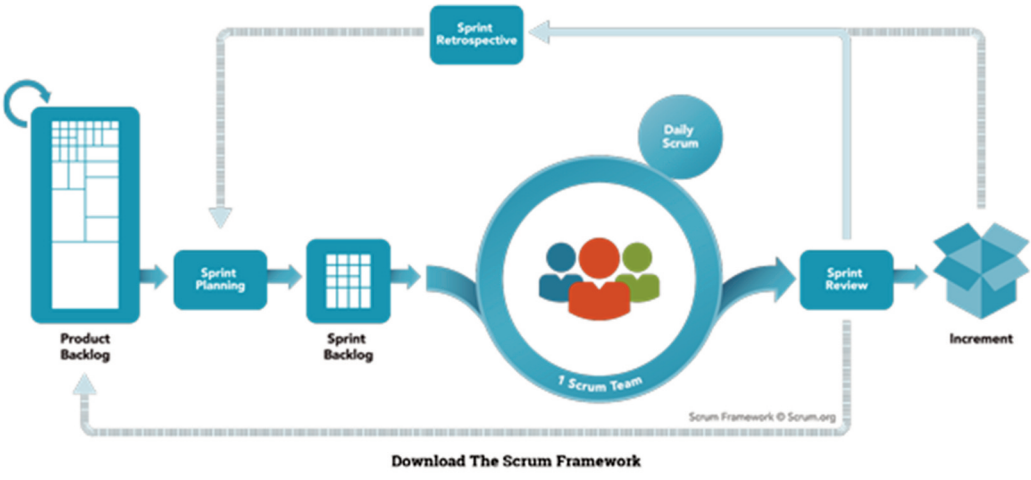

Fig. 5. Scrum framework (scrum.org).

Both Iterative and incremental development and Agile based processes are focused on feedback and continuous improvement of a product, as well as improvement in team collaboration and processes. Agile application for safety-critical systems development would improve management, quality (inspection) and provide better control, sustaining hands-on contact by the engineering team with software development and code writing. It may, as a result, be more cost-effective.

However, Agile frameworks do not address the notion of requirements traceability, focusing mostly on the delivery of functional features. As such, Agile methodology does not identify documentation creation as an essential part of the project, while it is a necessary part of the avionics product certification. Also, the typical rate of change of requirements in safety-critical domains, seems to be much lower than the rate of change seen in other industries. For this reason, popular frameworks like Scrum, left unaltered, seem to have limited application for the development of safety-critical aviation products.

AGILE APPLICATION FOR SAFETY CRITICAL SOFTWARE DEVELOPMENT

Agile vs DO-178C

The DO-178C standard does not define the process of software development, but objectives to be met. The standard explicitly allows iterative development processes, and requires the creation of detailed documentation and assurance of full requirements traceability.

In this context, creation and implementation of an Agile framework that addresses all safety-critical software limitations and fully supports the DO-178C standard would be a turning point for the industry. An Agile framework that could be successfully implemented for the development of safety-critical software would allow for reduced time and cost of new product development, which is a key constraint for the introduction of new avionics technologies.

A review of the potential constraints of Agile application may allow us to compile a list of challenges to be solved by the framework and to recognize major obstacles, whether they are related to formal requirements, customers, organization and culture, or internal fear of failure.

Big engineering organization culture.

Adoption of Agile can be challenging from the perspective of a big industrial company. These companies often suffer from a rigid organizational culture, with little ability to adapt. Implementing Agile is more of a cultural transformation than a process change. Cultural changes in this type of company are typically the most difficult type of change for leadership to execute. Agile methods would require changes from individuals, teams and top-level management at the same time. The typical culture of a big industrial company is derived from the waterfall and manufacturing approach. In such an environment Agile adoption may be deviated from by application of Agile terminology to existing rigid processes and adding Agile-like activities without any value to the project. In this paper the authors focus on the product development processes required for safety-critical system development. Cultural changes are not the subject of the paper.

Size of development team

No matter the size of the project team, the main challenge is to keep consistent system architecture throughout the iterations of product development. Starting Agile development from the moment when the system architecture is defined may be an option, however the advantage of changes being more straightforward to implement may be partially lost. Another potential obstacle that needs to be considered is the planning process of DO-178C. The challenge is how to keep the Software Development Plan (SDP), and Plan for Software Aspects of Certification (PSAC) up to date with incremental and iterative development of the product and its architecture. A strategy to keep these processes out of the Agile

framework may work, but it is questionable if we can build a proper definition of 'done' in this case. Smart integration of the planning process with Agile software development may be feasible, with support from dedicated teamwork tools. One possible approach is to include planning and documentation creation tasks to the development process.

Scalability of the framework

Looking at Agile from a scalability point of view within an organization, the independency requirement for DAL A and B may be difficult to address, unless there is an effective distribution of work allocation for teams involved in the project.

According to DO-178C the phrase „with independence" refers to a separation of responsibilities, where the objectivity of the verification and validation processes is ensured by virtue of their „independence" from the software development team. For objectives that must be satisfied with independence, the person verifying the item (such as a requirement or source code) may not be the person who authored the item, and this separation must be clearly documented. In some cases, an automated tool may be equivalent to independence. However, the tool itself must then be qualified if it substitutes for human review. Such an "independence" requirement may potentially complicate the allocation of work to different teams working in a project. Assuming a framework that would work in similar way to a scaled Scrum, there should be teams cross-checking the work content that other teams accomplished in the previous sprint, as a part of their next sprint. It may be difficult to manage, and any potential Agile benefits could be consumed by the complexity of the work organization.

An analysis of the occurrence of the various software levels on the avionics market may help us to understand what proportion of projects could be addressed using an Agile framework, whilst considering appropriate DO-178C objectives. According to references, more than 60% of the software in Avionics is level C or D. This means that building an efficient Agile framework that works up to level C would address most of the market needs. As the requirement of satisfying DO-178C objectives with independence starts from level B, such a framework could be applied to an organization and its development teams, exactly as it is defined by Scrum. From scalability point of view Agile framework that works up to level C would address most of the market needs. New framework allows to apply development team's organization exactly as it is defined by Scrum. Development team members can review the features internally in the team, without needing to engage with external resources.

CONCLUSIONS

The software in avionics is typically safety-critical, so certain standards like DO-178C need to be followed to successfully certify the system. There is a significant difference between the approaches to safety-critical and non-safety-critical software development. Methodologies of running the projects are subject to many more dynamic changes in the non-safety critical industry. It drives faster product introduction and widely observed improvements in efficiency. Over the last 15 years Agile application has been the biggest game-changer from the development process point of view. Safety-critical software, due to the certification process, standards and complexity, is the area of the industry where implementation of new methods is more difficult and typically slower.

If we assume an Agile-type framework would produce a similar impact on safety-critical software in avionics, as seen in the non-safety-critical industry, that could be adjusted to be consistent with the certification processes and standards required, this would be a turning point for aerospace systems development. Such a solution could be feasible, by adopting a combination of the existing methods discussed above, namely the V-model and the iterative and incremental approach. This would need to be accompanied by the necessary cultural changes within the organization.

REFERENCES

[1] Sommerville, I., 2017, *Software Engineering*, 10th Edition, Pearson India. ISBN-13: 978-9332582699.
[2] Radio Technical Commission for Aeronautics RTCA Inc.
[3] SAE, 2010, Guidelines for Development of Civil Aircraft and Systems ARP4754A. SAE International.
[4] https://www.projectmanager.com/software/use-cases/waterfall-methodology
[5] Royce, W., 1970, „Managing the Development of Large Software Systems", *Proceedings of IEEE WESCON*, 26, pp. 328-388.
[6] Sheppard, G., 2017, Systems Engineering Management, The University of Warwick.
[7] Badiru, A. B., 2019, System Engineering Models: Theory, Methods, and Applications. Boca Raton, FL: CRC Press, Taylor & Francis Group.
[8] Brenchley, D., 2018, Requirements Management and High-Level Design. Warwick / Cheltenham: Warwick University (WMG) / GE Aviation.
[9] INCOSE, 2019, Z1 Issue 3.0. Retrieved June 3rd, 2020, from INCOSE Z Guides: https://incoseuk.org/Documents/zGuides/Z1_What_is_SE.pdf
[10] PMI, 2013, A Guide to the Project Management Body of Knowledge (PMBOK® Guide) (5th ed.). Newtown Square, PA, USA: Project Management Institute (PMI).
[11] Presland, I., 2019, System Engineering Management Module: System Engineering Lifecycles. The University of Warwick (WMG).
[12] SAE, 2010, Guidelines for Development of Civil Aircraft and Systems. SAE International.
[13] SEBoK(A), 2020, May 15th, SEBoK Glossary. Retrieved from SEBoK: Guide to the Systems Engineering Body of Knowledge: https://www.sebokwiki.org/wiki/Verification_(glossary)
[14] SEBoK(B), 2020, May 15th, SEBoK Glossary. Retrieved from SEBoK: Guide to the Systems Engineering Body of Knowledge: https://www.sebokwiki.org/wiki/Validation_(glossary)
[15] Walden, D., Roedler, G., Forsberg, K., Douglas, H., and Shortell, T., 2015, INCOSE Systems Engineering Handbook: A Guide For System Life Cycle Processes and Activities (Fourth ed.). San Diego, CA: WILEY.
[16] Wasson, C. S. (2006). Systems Analysis, Design, and Development. New Jersey: WILEY.
[17] Scrum.org
[18] Beck, K., Grenning, J., Martin, R. C., Beedle, M., Highsmith, J., Mellor, S., van Bennekum, A., Hunt, A., Schwaber, K., Cockburn, A., Jeffries, R., Sutherland, J., Cunningham, W., Kern, J., Thomas, D., Fowler, M. and Marick, B., 2001, „Manifesto for Agile Software Development". Agile Alliance. Retrieved 14 June 2010.

METODY ROZWOJU OPROGRAMOWANIA KRYTYCZNEGO DLA BEZPIECZENSTWA W AWIONICE

Abstrakt

Artykuł podsumowuje metody rozwoju oprogramowania krytycznego dla bezpieczeństwa, oraz wpływ standardu DO-178C na potencjalne zastosowanie metod zwinnych. Wyjaśniamy w nim także kategoryzację oprogramowania krytycznego w systemach lotniczych. Artykul opisuje różnice w podejściu do procesu tworzenia oprogramowania od metody kaskadowej, przez model V, aż do metody iteracyjnej i przyrostowej, wraz ze wskazaniem ich największych zalet. Opisane zostały najważniejsze zasady leżące u podstaw metod zwinnych, oraz Scrum jako popularny framework stosowany w tworzeniu oprogramowania nie-krytycznego. Analiza możliwości zastosowania metod zwinnych do tworzenia oprogramowania krytycznego dla bezpieczeństwa w awionice została oparta na praktycznym

doświadczeniu autorów. Uwzględnia wymagania standardu DO-178C, wielkość projektu, skalowalność metody, oraz kulturę organizacji. Zdefniowanie zwinnej metody ramowej tworzenia oprogramownia krytycznego dla bezpieczeństwa, spójnej z procesem certyfikacji i istniejącymi standardami zostało uznane za potencjalny przełom dla rozwoju systemów awioniki.

Słowa kluczowe: Awionika, Oprogramowanie Krytyczne dla Bezpieczeństwa, Metody Zwinne, Agile, Inżynieria Systemowa.