Abstract—This paper summarizes the preliminary aspects of a doctoral research that has been conducted at the Brazilian Aeronautics Institute of Technology (ITA). This research has the objective of developing the CARD-RM, a Certifiable, Agile, Reusable, and Disciplined Reference Model for airborne software. It aims to define a generic model that can be instantiated in each airborne software project, integrating agile practices, in order to improve efficiency without interference in DO-178C compliance.

Keywords-component; software; agile; reference model; certification; safety-critical; DO-178C (key words)

I. INTRODUCTION

The development of safety-critical software is usually part of a regulated environment. A software development error can directly cause losses of human lives or has other catastrophic consequences. Examples include systems that control aircrafts, nuclear reactors, and medical devices. The correctness of such software needs to be demonstrated with high assurance.

Regulatory agencies in safety-critical industries require product, including the embedded software, to meet stringent certification requirements, as DO-178C [1] in aviation.

Software development projects, for aeronautical applications and systems, have particular characteristics than software projects in other domains. This kind of embedded software poses additional challenges, due to high dependability of software components and complex integration.

Airborne software projects have long cycles of development and verification. According to Marques and Dias [2], this kind of software project usually demands three to four years.

A Reference Model (RM) is intended to provide a higher level of commonality, with definitions that should apply to all software projects. Additionally, it is an abstract framework for understanding significant relationships among the entities of some environment, using consistent standards or specifications supporting that environment [3].

Nowadays, there are many reference models available: OASIS [3], CMMI [4], and MPS.BR [5]. They typically do not foresee the compliance with regulations (norms and standards) or provide traceability.

The authors of this paper believe that a RM provides two natural and intrinsic characteristics: reusability and discipline. In the scope of this doctorate research, other characteristics are included in the proposed RM: certifiability and agility.

According to Leite et al. [6], software reuse has been a goal for Software Engineering (SE) researches and practices, as a means to reduce development costs and improve quality. In SE, there are two types of software reuse: software product reuse and software process reuse. The software product reuse may cause the propagation of software errors from original products to derived products. The software process reuse causes continuous improvement and identifies the best practices that can be captured as part of a reusable framework.

According to Boehm and Turner [7], "discipline is the foundation for any successful endeavor. Discipline creates well-organized memories, stories, and experiences" and also "applies to memory and history agility to adjust to new environments, react, and adapt up, taking advantage of unexpected opportunities, and update experience basis for the future". It seems that there is intersection between reusability, discipline, and agility.

According to McMahon [8], there are results and experiences involving maturity models, such as the Capability Maturity Model Integration (CMMI) and agile methods.

Agile methods, as a set of best practices applied to the software development and verification, offer a collection of practices that allows constantly changes in the software product, seeking sustainable gains in productivity and quality of software development [9].

The purpose of the certification activity in aviation is to establish, through requirements, a minimum level of safety that must be checked in every aircraft and technically supervise the implementation of those requirements [10].

The certification must cover the design, manufacture, operation, and maintenance of aeronautical products. It is considered aeronautical product every aircraft, engine, or
propeller, and every component or part of them. An applicant is an aircraft manufacturer and he is responsible for obtaining the type design approval of an aeronautical product [10].

The DO-178C and other software standards have been demanding heavy documentation and are inspired in the waterfall methodology. The basic idea is that DO-178C is an evolution of DO-178 [11] (1982), DO-178A [12] (1985), and DO-178B [13] (1992). The DO-178 reviews have been used and recognized by the aviation certification authorities. The aeronautical community has defined these reviews through the Radio Technical Commission for Aeronautics (RTCA) committee. This was done to promote their reviews, using worldwide certification agencies, aviation industry, and academic representatives.

This paper summarizes the preliminary aspects of a doctoral research work that has been developed in the Brazilian Aeronautics Institute of Technology (ITA).

This research has the objective of developing the CARD-RM, a Certifiable, Agile, Reusable, and Disciplined Reference Model for airborne software. This Reference Model integrates agile practices and also fits its involved regulation in airborne software projects to improve efficiency and reusability, without interference in the DO-178C compliance.

The CARD-RM is a generic model that can be instantiated in each airborne software project. Its requirements are presented in Table I.

**Table I. CARD-RM Requirements**

<table>
<thead>
<tr>
<th>CARD-RM Goals</th>
<th>CARD-RM Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certifiable</td>
<td>The CARD-RM shall trace to DO-178C</td>
</tr>
<tr>
<td>Agile</td>
<td>The CARD-RM shall integrate Agile Practices defined in Agile Methods</td>
</tr>
<tr>
<td>Reusable</td>
<td>The CARD-RM shall be a generic model that can be instantiated in each airborne software project</td>
</tr>
<tr>
<td>Disciplined</td>
<td>The CARD-RM shall define a process that consider the continuous improvement</td>
</tr>
</tbody>
</table>

Table II presents the five phases of this doctoral research. This paper summarizes the Phase 1 and describes the CARD-RM definition. At Phase 1, the focus is the use of Scrum [14] and the compliance with DO-178C.

**Table II. Phases of the Doctoral Research**

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CARD-RM Definition</td>
</tr>
<tr>
<td>2</td>
<td>CARD-RM Activities and Goals Definition</td>
</tr>
<tr>
<td>3</td>
<td>CARD-RM Instantiation in a Software Project</td>
</tr>
<tr>
<td>4</td>
<td>Software Planning Audit for Compliance</td>
</tr>
<tr>
<td>5</td>
<td>Discussion of Results</td>
</tr>
</tbody>
</table>

Beyond of this section, this paper has more four sections. Section II discusses the background and Section III provides the Scrum basics. The CARD-RM is presented in Section IV, the next steps are summarized in Section V and main conclusions are presented in Section VI.

**II. BACKGROUND**

**A. DO-178C Environment**

The DO-178C establishes considerations for developers, installers, and users when designing an embedded equipment using software. This norm defines five software levels. Each level of software was defined in terms of objectives that must be achieved to approve the embedded software as part of aircraft certifications. Among the five software levels (A, B, C, D, and E), level A is the most rigorous.

The 71 DO-178C objectives are presented in 10 tables, published in Annex A of the norm. The CARD-RM is focused only from Table A-1 to A-7.

**B. Related Works**

The idea of reconciling agile development and discipline presented in conformant processes, especially in regulated environments, is not new [7].

Although the software development for aeronautical applications has particular details, this area needs the increasing of competitiveness, basically because this industry needs to deliver software with additional functionalities at lower costs. The new methods and visions need to be considered without interfering the compliance regulation.

A variety of software industries are applying agile methods or agile principles that are compatible with the specific needs of an organization. Specific efforts, in balancing agility and discipline, have been generated [15][17][18][20].

According to Stalhane and Hanssen [15], the ISO 9001 [16] was mapped to agile development principles and some differences were presented. According to McMahon [8], it is possible to integrate agile software development with CMMI practices. The possibility of applying agile software development with ECSS standards was reported in [17]. Vuori, in [18], has defined the mapping of IEC 61508 [19] and recommended the development tasks to a simplified agile process model.

The issues from the perspective of aerospace industry and the mapping from agile practices to DO-178B were presented in [20]. From that work, DO-178B objectives were divided into three categories: fully compatible agile practices, easily compatible agile practices, and problematic agile practices. Their results have only considered the agile methods of XP, Scrum, and Crystal. At that work it was presented only “what” to map, but not “how” to integrate practices to DO-178B process. Additionally, that work only has considered the DO-178B, not the DO-178C.

**C. Agile Methods**

According to Vuori [18], there is a tendency for companies to transform their software and product development practice into more incremental form, by using special agile software development. Although the incremental software development processes have been
previously used, agile principles are the main characteristics adopted within the following methods: Scrum [14], XP (eXtreme Programming) [21], Crystal [22], FDD (Feature Driven Development) [23], TDD (Test Driven Development) [24], and ASD (Adaptive Software Development) [25], among others.

There are the following key motivations for agile methods [26]:

- Early partial product;
- Better predictability;
- Final product better matching true client desires;
- Manageable complexity; and
- Early mitigation and discovery.

III. SCRUM BASICS

According to Jeff Sutherland [27], Scrum is an iterative and incremental framework for projects and products or applications development. The Scrum structures the development in cycles of work, also defined as Sprints.

One Sprint usually takes place one after the other and it has fixed duration, typically 2-4 weeks. It ends on a specific date, whether the work has been completed or not, and is never extended. Hence, they are time boxed.

A Scrum project is driven by a product vision compiled by the Product Owner, and expressed in a Product Backlog. The Product Backlog is a prioritized list of what is required, ranked in order of value to the customer or business, with the highest value items at the top of the list. It evolves over the lifetime of the project, and items are continuously added, removed, or reprioritized.

At the beginning of each Sprint, a Sprint Planning Meeting takes place. The Product Owner and Scrum Team review the Product Backlog, discuss the goals and context for the items, and the Scrum Team selects the items from the Product Backlog to commit to complete by the end of the Sprint, starting at the top of the Product Backlog.

Each item selected from the Product Backlog is designed and then broken down to a set of individual tasks. The list of tasks is recorded in a document called a Sprint Backlog.

After the Sprint ends, there is a Sprint Review, where the Scrum Team and the stakeholders inspect what was done during the Sprint, discuss it, and figure out what to do next. Present at this meeting are the Product Owner, the Team Members, and the Scrum Master, plus customers, stakeholders, experts, executives, and anyone else interested. Fig. 2 presents how a Scrum works.

The principles defined at a Scrum have been adapted in the CARD-RM definition. Due to the reconciling of the certification and agility, the reference model adopts a modified definition of the Scrum that will be presented in next sections of this paper.

IV. CARD-RM DEFINITION

According to the DO-178C, a software life cycle is “an ordered collection of processes determined by an organization to be sufficient and adequate to produce a software product”. According to Shumate and Keller [28], a process is a “set of steps that must be accomplished to complete a task”.

As previously mentioned, the CARD-RM have been developed as part of a doctoral research.

This paper uses the flow presented in Fig. 1 [27], and provides an adaptation to achieve the airborne system and software development and certification is presented in Fig. 2. Table III provides the correlation between the Scrum Definition and the proposed CARD-RM.

![Figure 1. How the Scrum Works [27]](image)

<table>
<thead>
<tr>
<th>Scrum Definition</th>
<th>CARD-RM Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Owner</td>
<td>Applicant</td>
</tr>
<tr>
<td>Vision Document</td>
<td>System Basic Data</td>
</tr>
<tr>
<td>Product Backlog</td>
<td>System Backlog Requirements</td>
</tr>
<tr>
<td>Sprint Backlog Features</td>
<td>Software High-Level Requirements</td>
</tr>
<tr>
<td>Sprint Backlog Tasks</td>
<td>Software Design Data</td>
</tr>
</tbody>
</table>

As previously mentioned, an aircraft is subject to a certification process. However, all the systems that are part of the aircraft shall achieve the complete certification basis at the end of the development and test campaign.

The aircraft certification tests start before the full compliance of the objectives defined in DO-178C. The Software Under System Test (SUST) must have maturity and representativeness for an aircraft testing. They must be achieved and changed, according to the type of testing and credit for the aircraft certification campaign.

This paper defines three types of compliance that can be achieved during the development of software products to be delivered. Each Sprint will deliver, at its end, a software
product that is adherent, according to the compliance levels defined in Table IV.

### TABLE IV. COMPLIANCE LEVEL AND SUST TYPES

<table>
<thead>
<tr>
<th>Sprint Compliance Level</th>
<th>Type of Aircraft Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint Compliance Level 1 (SCL1)</td>
<td>Ground and Laboratory Tests</td>
</tr>
<tr>
<td>Sprint Compliance Level 2 (SCL2)</td>
<td>Development Flight Tests</td>
</tr>
<tr>
<td>Sprint Compliance Level 3 (SCL3)</td>
<td>Certification Flight Tests</td>
</tr>
</tbody>
</table>

The proposed Sprint Compliance Level 1 (SCL1) establishes that the functions required for a given delivery are present and software high-level requirements are verified. The proposed SCL2 establishes that the functionalities of the system are complete and verified. The SCL3 establishes functionalities are in full compliance with all DO-178C objectives.

### A. System Basic Data

The applicant defines the system definition and requirements. During the design of the airborne system, typically, a set of basic assumptions of the system is defined on contracts of the system supplier. This set of assumptions represents the basic requirements that a system must comply.

These basic requirements are included in the System Basic Data (SBD). In the CARD-RM, the product owner is the applicant, and the SBD represents the equivalent of Vision Document defined in Scrum.

Normally, for small projects, the vision of the product is simplified. Just like the Vision Document, in this adaptation, the SBD does not change throughout the system life cycle and it is part of the contract between the applicant and the software supplier.

Although agile methods preconize that changing requirements are welcome, even late in development, in the scope of this research, the focus in use of agile are concentrated in continuous integration and early partial product.

### B. System Backlog Requirements

Once defined the SBD, the next step is the detailment and composition of system requirements. Typically, system requirements define the functionality of the system that has been developed.

An airborne system is not simply composed of software. Several systems are comprised of software, electronic hardware, and other parts (mechanical, pneumatics, and hydraulics). Not all system features are directly applied to the software.

The System Requirements Backlog needs two refinements. Initially, the System Requirements Backlog should define all system features, generating the System Requirements Backlog with all features. On the second refinement, System Requirements Backlog must present the system features applicable to software, generating the System Requirements Backlog (SW features). The system features applicable to other non-software parts are not in the scope of this paper.

### C. Planning

The DO-178C defines 5 software plans:
- Plan for Software Aspects of Certification (PSAC);
- Software Development Plan (SDP);
- Software Verification Plan (SVP);
- Software Configuration Management Plan (SCMP); and
- Software Quality Assurance Plan (SQAP).

Each plan should address “what”, “how”, “when”, and “who” performs the activities defined to meet all the DO-178C objectives, for a desired software level.

Plans must instantiate the CARD-RM presented in the next sections, achieving the DO-178C objectives presented in Table A-1.

At this paper, the Software Requirements Standards, the Software Development Standards, and the Software Coding Standards, as required by DO-178C Sections 11.6, 11.7, and 11.8 are considered included in the Software Development Plan (SDP).

### D. Sprint Planning Meeting 1 (SPM 1)

The DO-178C defines two levels of software requirements. The Software High-Level Requirements (SW-HLR) usually represents “what” is to be designed and the Software Low-Level Requirements (SW-LLR) represents “how” to carry out the design.

During the SPM 1, the focus of the software team is to understand the System Requirements Backlog (SW features), to create the SW-HLR, and to address the DO-178C (Table A-2, objectives 1 and 2).

This SW-HLR should include functional, performance, and safety-related requirements. Table V presents the information that should be addressed and not addressed by SW-HLR.

### E. Sprint Planning Meeting 2 (SPM 2)

During the SPM 2, the focus of the software team is to understand the Software High-Level Requirements for the Sprint and to create the Software Design Data for the Sprint, which includes SW-LLR and software architecture, to address the DO-178C (Table A-2, objectives 3 and 4). The Software Design Data is used to implement source code. Table VI presents the characteristics of the Software.

### TABLE V. SW-HLRS INFORMATION

<table>
<thead>
<tr>
<th>Should Address</th>
<th>Should Not Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Functionality</td>
<td>• Partitioning software into modules</td>
</tr>
<tr>
<td>• External interfaces</td>
<td>• Allocating functions into modules</td>
</tr>
<tr>
<td>• Performance</td>
<td>• Flow of information or control among modules</td>
</tr>
<tr>
<td>• Quality attributes</td>
<td>• Design constraints</td>
</tr>
<tr>
<td>• Safety</td>
<td>• Safety</td>
</tr>
</tbody>
</table>

276
Design Data.

TABLE VI. SOFTWARE DESIGN DATA INFORMATION

<table>
<thead>
<tr>
<th>SW-LLR Characteristics</th>
<th>Software Architecture Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>• More detailed than SW- HLR</td>
<td>• Shows physical partitioning of the system into components and data flow between them</td>
</tr>
<tr>
<td>• Trace to SW-HLR</td>
<td>• Functional processes are allocated to physical units</td>
</tr>
<tr>
<td>• Should be sufficient for coding</td>
<td>• Control flow (process activation)</td>
</tr>
</tbody>
</table>

F. Sprint Implementation

The Sprint Implementation can be defined in two parts, development and formal verification. During the development, the Source Code is generated, by using the Software Design Data.

As soon as the source code has been implemented, the Software Integration generates the executable code. The generation of the Source and Executable Code addresses the DO-178C (Table A-2, objectives 5 and 6).

The DO-178C defines that some objectives must be accomplished with independence. Although independence is required for some activities, these can be informally executed without independence, not for credit of compliance, but for preliminary errors removal.

The agile methods emphasize testing, during the development. As DO-178C separates into two dedicated phases and the most of testing activities requires independence, in a first moment, it appears that the testing, during development, is prohibited.

Informal software testing can be performed, during the development, but not for credit of compliance. This activity anticipates the errors introduced during the source code generation. Fig. 3 presents the Sprint Implementation flow.

Some benefits of the testing are:

• Errors in Source Code are anticipated;
• Test Cases and Procedures will acquire maturity and can be reviewed, by using the results of testing;
• Predictability of the test results; and
• Everyone is a Developer and Informal Tester.

As the DO-178C requires independence for some objectives, informal tests cannot be used for credit, during a Level A of software development. After the completion of the source code that implements the Software Design Data for a given sprint, the formal verification can be performed to achieve DO-178C verification objectives.

G. Formal Verification

The purpose of the Formal Verification is to detect and report errors that may have been introduced. The Software Verification Process is typically a combination of reviews, analyses, and tests.

Typically, an output of each review represents a list of action items, and the appropriate completed checklists. The review is complete when: (i) all issues and action items identified, during the review, have been closed; (ii) the associated Problem Reports are generated; and (iii) all outputs have been put under configuration control.

An analysis provides an assessment of accuracy, correctness, and completeness. It may examine in detail: functionality, performance, traceability, and safety implications of a software component relationship to other components within the airborne system. The analysis methods provide repeatable evidence of correctness.

For testing, the Requirements-Based Tests (RBT) is mandatory, by DO-178C, to test Software High-Level and Low-Level Requirements. The formal verification activities to be performed in each Sprint Compliance Level are presented in Table VII.

V. NEXT STEPS

According to Table 1, the next steps are comprised of another 4 phases. The next phase (Phase 2) will detail the CARD-RM (Certifiable, Agile, Reusable, and Disciplined Reference Model) in terms of generic activities and goals. At this phase, other Agile Methods such as Test-Driven Development (TDD), Feature-Driven Development (FDD), and Adaptive Software Development (ASD) will be considered. Additionally, the use of Model-Driven Development (MDD) will be included.

During the Phase 3, the CARD-RM will be instantiated in a Level A software project. At this point, the Reference Model will be tailored to a software project and the project plans, mentioned in Section IV - C of this paper, will be released.

Phase 4 will be a validation phase of the plans released in Phase 3. The FAA Job Aid [29] will be used to evaluate the compliance with DO-178C. This phase will provide the assurance that the CARD-RM applied to a software project is certifiable.

Phase 5 will be a validation of the CARD-RM, in terms of Agility, Reusability, and Disciplined Reference Model. These three attributes of the CARD-RM will be analyzed as part of the discussion and results in a real software project. The reusability will be investigated, by using the principles defined in [30], and integrated in the CARD-RM model.
TABLE VII. FORMAL VERIFICATION ACTIVITIES AND OBJECTIVES BY COMPLIANCE LEVEL

<table>
<thead>
<tr>
<th>Activity</th>
<th>Verification Method</th>
<th>DO-178C Objectives Covered</th>
<th>SCL 1</th>
<th>SCL 2</th>
<th>SCL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review of Software Requirements</td>
<td>Review</td>
<td>Table A-3</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Review of Software Design</td>
<td>Review</td>
<td>Table A-4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Review of Source Code</td>
<td>Review</td>
<td>Table A-5 (obj. 1-6)</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Review of Software Integration</td>
<td>Review</td>
<td>Table A-5 (obj. 7)</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Review of Parameter Data Item</td>
<td>Review</td>
<td>Table A-5 (obj. 8 and 9)</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Review of Test Cases and Procedures (SW-HLR)</td>
<td>Review</td>
<td>Table A-7 (obj. 1)</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Review of Test Results (SW-HLR)</td>
<td>Review</td>
<td>Table A-7 (obj. 2 and 3)</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Review of Test Results (SW-LLR)</td>
<td>Review</td>
<td>Table A-7 (obj. 2 and 4)</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Test Execution (SW-HLR)</td>
<td>Testing</td>
<td>Table A-6 (obj. 1, 2 and 5)</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Test Execution (SW-LLR)</td>
<td>Testing</td>
<td>Table A-6 (obj. 3, 4)</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Structural Coverage</td>
<td>Analysis</td>
<td>Table A-7 (obj. 5-8)</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Source to Code Traceability</td>
<td>Analysis</td>
<td>Table A-7 (obj. 9)</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
VI. CONCLUSION

This paper has summarized the preliminary aspects of a doctoral research that has been conducted at the Brazilian Aeronautics Institute of Technology (ITA).

The research has the objective of developing the CARD-RM, a Certifiable, Agile, Reusable, and Disciplined Reference Model for airborne software.

This Reference Model consists of a set of unifying concepts and relationships within a specific problem domain for airborne software certification, and it is independent of technologies, implementations, or even other concrete details.

The key motivation for agile methods identified in [26], were included in the Reference Model. During this process, early partial product should be delivered at the end of each Sprint. The software product should be developed with better predictability, as informal testing is conducted jointly with source code development. The errors should be discovered in better opportunities and early mitigation could be introduced. However, formal testing should be performed during the Formal Verification.

The final product should better match the applicant desires through the refinement of a System Requirements Backlog.

Due to the increasing complexity of airborne systems, the authors of this paper recommend that approaches like CARD-RM be used and/or adapted, aiming to integrate agile methods and their best practices to enhance efficiency while maintaining DO-178C compliance.

For future works, it is suggested the implementation of the next four phases of this research.

REFERENCES


