AN ANALYSIS OF SOFTWARE REQUIREMENTS SPECIFICATION CHARACTERISTICS IN REGULATED ENVIRONMENTS

Johnny Marques and Sarasuaty Yelisetty

Computer Science Division, Aeronautics Institute of Technology,
Sao Jose dos Campos, Brazil

ABSTRACT

Requirements Engineering is the set of activities involved in creation, managing, documenting, and maintaining a requirements' set for a product. Engineering involves the use of systematic repeatability techniques to ensure that the Software Requirements are complete, consistent, valid, and verifiable. Software Requirements Specification is an organized process oriented toward defining, documenting and maintaining requirements throughout the development life cycle. Many authors suggest that requirements should always focus their claims on what the software product needs to address, without specifying how to implement them. However, the detail of Software Requirements is influenced by several factors such as: organizational thinking; existing specification standards; and regulatory needs. This work fits exactly with regulatory needs, where the characteristics of Software Requirements Specification in Regulated Environments such as aeronautics, railways and medical are presented and explored. This paper presents and analysis of software requirements specification characteristics in regulated environments. The four characteristics identified are: consistency (internal and external), unambiguity, verifiability, and traceability. The paper also describes the three standards used in these regulated environments (RTCA DO-178C, IEC 62279 and IEC 62304) and examines their similarities and differences from a Requirements Specification standpoint. The similarities and differences will be used to address a future requirements framework universal process that can be configured to address each standard by the usage of Software Process Lines.

KEYWORDS

Software, Requirements, Certification, Standards

1. INTRODUCTION

Typically, Safety-Critical Software is developed in environments regulated by standards and Regulatory agencies in safety-critical industries typically require system providers to meet stringent certification requirements [1]. The development of safety-critical systems is usually part of a regulated environment. A software development error can directly cause losses of human lives or has other catastrophic consequences [2]. Examples are found in domains such as aviation, automotive, medical, railway, space, and nuclear. In this work, the software in these domains is defined as Software in Regulated Environments (SRE). A common characteristic in the rules and standards of these domains is the Requirements Specification. The literature has addressed the various issues in Requirements Specification, which may involve incomplete, incorrect, ambiguous, conflicting, or inconsistent requirements [3][4][5]. SRE does not involve a completely heterogeneous area, but consists of many different development cultures, which have common characteristics that allow them to be correlated, such as: a) Software product type; b) The role of software in the system; c) The size of the system; and d) The level of risk of the system.

DOI: 10.5121/ijsea.2019.10601

Standards published by committees, international technical entities, or regulatory agencies influence the development of SRE by means of guidelines for software processes and products [6], given the risk. The aim of this paper is to provide an analysis of software requirements specification characteristics in regulated environments. The specific objectives of this paper are: a) Present the selected software standards (RTCA DO-178C [7], IEC 62279 [8] and IEC 62304 [9]); and b) Identify their similarities and differences in Requirements Specification.

In addition to section 1, the paper includes another 3 sections. Section 2 briefly describes the three standards that are part of the scope of this paper: RTCA DO-178C, IEC 62279 and IEC 62304. Section 3 presents the similarities and differences. Section 4 presents the conclusion.

2. REGULATORY SOFTWARE STANDARDS

Regulated environments are those that bring impacts to society in general. Therefore, they need standards for regulation of products and services delivered by companies. Society expects to receive safe and reliable services and products. In all the various regulated environments, such as: aeronautics, railway, automotive, nuclear, medical, military, among others, there are many standards that cover various technologies, including software development. As a direct consequence, specific rules must demonstrate that products and services are safe and reliable for operation. The software safety standards usually define objectives or goals that must be accomplished by the software project [10].

2.1. Selection of Software Standards

There are many standards for SRE. We identified 7 attributes normally present in Requirements Specification, aiming to establish the criteria for comparison and differentiation of the standards. The 7 attributes are:

- At1 - Traceability between Software Requirements and System Requirements;
- At2 - Traceability between Test Cases and Software Requirements;
- At3 - Description of Software Requirements with expected performance;
- At4 - Software Requirements compatibility with the computing environment;
- At5 - Description of Software Requirements in terms of interfaces with others Software and/or systems;
- At6 - Consistency between Software Requirements; and
- At7 - Software Requirements allocation in Software Architecture.

The choice of the seven attributes listed above is justified in [11]. Seven standards were chosen, due to their representativeness in their regulated environments, for satisfaction analysis of the 7 attributes: RTCA DO-178C [7], IEC 62279 [8], IEC 62304 [9], RTCA DO-278A [12], ISO 26262-6 [13], ECSS-E-ST-40C [14] and IAEA SSG-39 [15]. Table 1 presents the results of this satisfaction analysis assessment of the 7 attributes.

Each attribute has been rated 0, 1, or 2. Grade 0 indicates that the standard does not have or mention the attribute. Grade 1 indicates the attribute is mentioned but does not consider it mandatory and needs to be evaluated to comply with any objective or activity of the standard. In grade 1, the attribute may be mentioned as an example within the text of the standard, but it is not explicitly required. Finally, grade 2 indicates that the standard requires an explicit activity or objective indicated by the attribute. Thus, the attribute is considered a requirement within the evaluated standard.

Table 1. Evaluation of Software Standards in Regulated Environments

Standard	Environment	At1	At2	At3	At4	At5	At6	At7	Total
RTCA DO-178C [7]	Aeronautical	2	2	1	2	1	2	2	12
IEC 62279 [8]	Railway	2	2	2	2	1	2	2	13
IEC 62304 [8]	Medical	2	2	2	1	2	1	2	12
RTCA DO-278A [12]	Air Traffic Management	2	2	1	2	1	2	2	12
ISO 26262-6 [13]	Automotive	1	2	2	1	1	2	1	10
ECSS-E-ST-40C [14]	Aerospace	1	2	1	1	1	0	1	7
IAEA SSG-39 [15]	Nuclear	0	1	1	1	1	1	1	6

Due to the similarity between RTCA DO-178C and DO-278A, they presented identical evaluations regarding the attributes. Both were defined by the same SC-205 committee, where RTCA DO-178C focuses on the development of embedded aeronautical systems and RTCA DO-278A focuses on aeronautical ground support systems such as communication, navigation and surveillance for air traffic control. Based on the results consolidated in the last column of Table 1, which reflects the sum of the score obtained for all attributes defined by the authors of this work, the RTCA DO-178C, IEC 62279, and IEC 62304 were the standards that most adhered to the attributes present in Requirements Specification.

Munch et. al [1] consider that the number of organizations that need to verify compliance with regulatory standards is increasing. Many of these standards-based regulations require the presence of explicit software development processes. Therefore, the activities performed should have repeatability and traceability within the proposed software development process.

These standards have objectives or activities that must be satisfied for the software product to be approved for operation in its environment. Regulatory agencies, or other entities, usually require adherence to established standards, such as IEC 62279 [8] for the railway domain, DO-178C [7] in aeronautics and IEC 62304 [9] in the medical field.

2.2. RTCA DO-178C

The rapid increase in the use of the software in aircraft has resulted in the need for an industry-accepted software development guidance to meet airworthiness requirements. The RTCA DO-178C exists to satisfy this need. This document provides the aeronautical community with guidelines on the software development processes that the onboard systems and equipment need to demonstrate compliance. This is a Document (DO) established by the Radio Technical Commission for Aeronautics (RTCA). The RTCA DO-178C [2] is an evolution of DO-178 (1982), DO-178A (1985) and DO-178B (1992).

Over the years, the Federal Aviation Administration (FAA) and the European Aviation Safety Agency (EASA) have recognized DO-178 revisions as an acceptable means of developing aeronautical software. All versions have been defined by representatives of the aeronautical community affiliated to the RTCA. The DO-178C establishes considerations for developers, installers, and users when designing an embedded equipment using software [16].

FAA AC 20-115D [17] currently recognizes RTCA DO-178C as an acceptable method for approving systems and/or equipment using software. Each of its 5 levels of software contains into

objectives that must be satisfied to enable it to be approved as part of an aircraft certification process. Of the five existing Software levels (A, B, C, D, and E), Level A is more stringent and requires compliance with all objectives of the standard. Level E refers to software products whose malfunction affect safety margins. ARP 4754A [18] classifies each system failure with an associated criticality into five categories. Thus, the failure condition classification is associated with levels defined in the RTCA DO-178C, according to Table 2 and the satisfaction of a set of associated objectives becomes necessary. The authors used the number of associated objectives from [19] and counted along the standard the number of objectives that require independence.

The DO-178C's 71 objectives are organized into 10 specific objective tables within the standard. Figure 1 presents an overview of the organization of these tables. As part of the RTCA DO-178C release effort, other supplementary standards have been developed, including special recommendations on uses of: tool qualification (RTCA DO-330 [20]), model-based development (RTCA DO- 331 [21]), object-oriented technology (RTCA DO-332 [22]) and formal methods (RTCA DO-333 [23]).

Table 2. System failure conditions and associated objectives

System Failure Condition	**Required Software Level**	**Number of Associated Objectives [19]**	**Number of Associated Objectives with Required Independence**
Catastrophic	A	71	31
Hazardous	B	69	19
Major	C	62	5
Minor	D	26	2
No Safety Effect	E	0	0

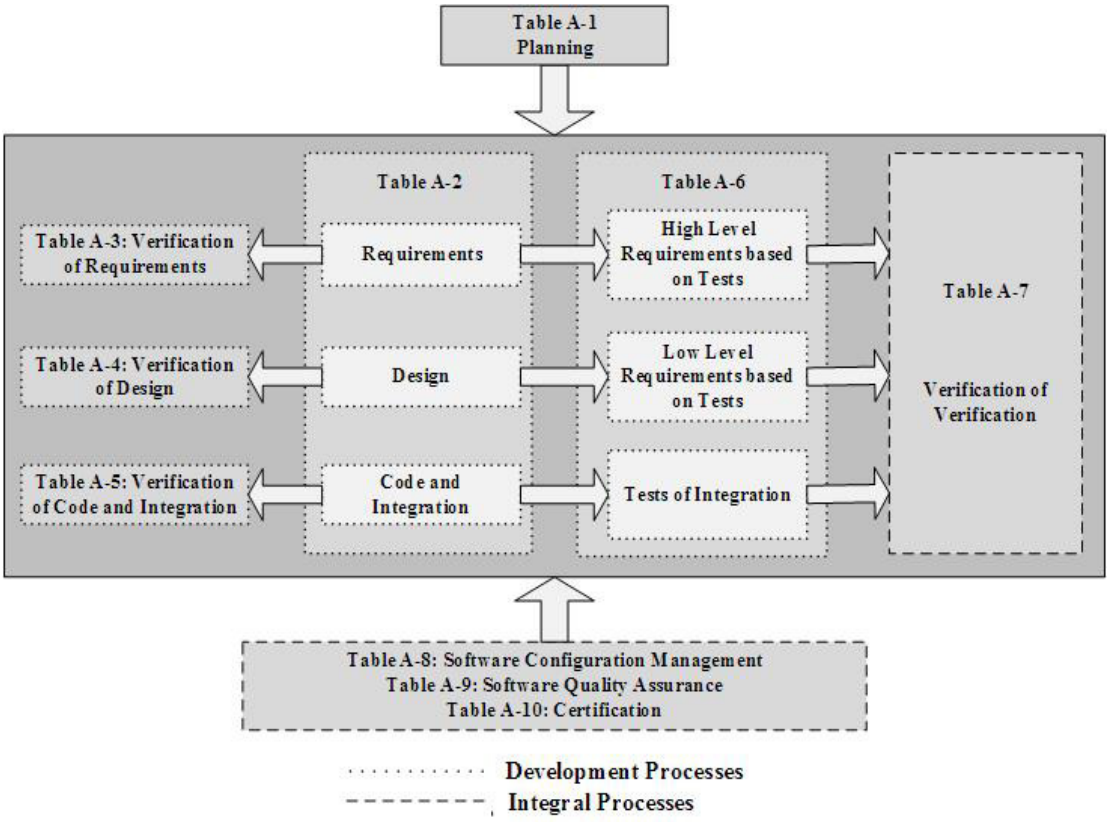

Figure. 1. RTCA DO-178C tables' organization [24]

The RTCA DO-178C has a significant number of objectives associated with Software Requirements development, using as input System Requirements that will be implemented by Software. There are two levels of Software Requirements on RTCA DO-178C. Software High-Level Requirements (SW-HLR) generally represent "what" should be designed. SW-HLRs include functional, performance, interface and safety related requirements. The Software Low-Level Requirements (SW-LLR) represent the how-to, providing details on implementing Software in code [25]. SW-LLRs include the features required for source code development, such as data coupling and control features.

The rationale for two levels of Software Requirements is the need to provide traceability and refinement from System Requirements to the level of implementation in source code. RTCA DO-178C requires the definition of a Software Requirements Standards (SRSt) which shall define the methods, notations, rules and tools to be used to develop the SW-HLRs, which shall be adherent to SRSt.

Activities associated with the development of High-Level Software Requirements include:

1. Each allocated System Requirement for Software must be specified in Software High-Level Requirements (RTCA DO-178C Section 5.1.2 (c));

2. Each Software High-Level Requirement must adhere to the Software Requirements Standards (SRSt) and be verifiable and consistent (RTCA DO-178C Section 5.1.2 (e));

3. Each Software High-Level Requirement shall be established in quantitative terms with tolerances, where applicable (RTCA DO-178C Section 5.1.2 (f)); and

4. Each derived Software High-Level Requirement must have a justifiable reason for its existence (RTCA DO-178C Section 5.1.2 (h)).

The High-Level Software Requirements review should ensure that they are:

1. Traceable and compliant with System Requirements (RTCA DO-178C Section 6.3.1 (a) and (f));

2. Accurate and consistent (RTCA DO-178C Section 6.3.1 (b));

3. Compatible with the computer environment (RTCA DO-178C Section 6.3.1 (c));

4. Verifiable, possible to provide an evidence of satisfaction (RTCA DO-178C Section 6.3.1 (d)); and

5. Adhering to Software Requirements Standards (SRSt) (RTCA DO-178C Section 6.3.1 (e)).

The Software architecture is developed from the Software High-Level Requirements (DO-178C Section 5.2.1 a). Additionally, the manufacturer shall develop and document the architecture, including the interfaces between internal and external components (DO-178C Section 5.2.2d).

2.3. IEC 62279

The software is widely used in the rail system such as train propulsion system, brake system, train control system, train detection system and driver display unit [26]. When developing software in the rail sector, IEC 62279 is the most common standard to be followed in terms of RAMS (Reliability, Availability, Maintenance and Safety) [26]. IEC 62279 is a standard that regulate the development, deployment and maintenance of software safety systems for railway applications. It contains requirements of the developing organization (roles and competencies), life cycle (phases, documentation and methods) and software assurance (testing, verification, validation and quality assurance and evaluation) [27]. IEC 62279 requires manufacturers to assign a Safety Integration Level (SIL) for systems with Software. This classification is based on the potential danger that could result in harm to the user in case of abnormal system behaviour. The SIL concept involves a class of safety requirements for functions, systems, subsystems or components. A SIL consists of two factors:

1. A range of values for a Tolerable Hazard Rate (THR); and
2. Measures to be implemented in the project during the development process.

A SIL can be assigned to any relevant safety function, system, subsystem, or component that can be categorized into five distinct levels from 0 to 4. SIL 4 is most rigorous and requires all mandatory activities to be performed. SIL 4 is equivalent to SIL 3 and SIL 2 is equivalent to SIL 1, however, SIL 2 and 4 have activities that require independence in their execution. The total activities of each SIL are presented in Table 3. The activities were counted along the standard. IEC 62279 has 11 phases: Planning, System Development, Requirements, Architecture Design, Component Design, Implementation, Testing, Integration, Validation, Maintenance and Evaluation.

Table 3. Safety Integration Levels by IEC 62279 and associated activities

SIL	Tolerable Hazard Rate (THR) [28]	Mandatory Activities	Highly Recommended Activities	Recommended Activities
0	Not Applicable	4	16	53
1	10^{-5}=<THR<10^{-6}	5	54	53
2	10^{-6}=<THR<10^{-7}	5	54	53
3	10^{-7}=<THR<10^{-8}	19	84	69
4	10^{-8}=<THR<10^{-9}	19	84	69

Figure 2 provides an overview of the life cycle of IEC 62279. It has a large set of activities associated with the development of Software Requirements described throughout the standard. The Software Requirements Specification phase should express the properties of the software to be developed, including its functionality, robustness, maintainability, efficiency, usability and portability (IEC 62279 Section 7.2.4.2). Additionally, the Software Requirements Specification should be structured to ensure that it is complete, clear, accurate, verifiable, testable and traceable to the inputs used in its definition (IEC 62279 Section 7.2.4.4).

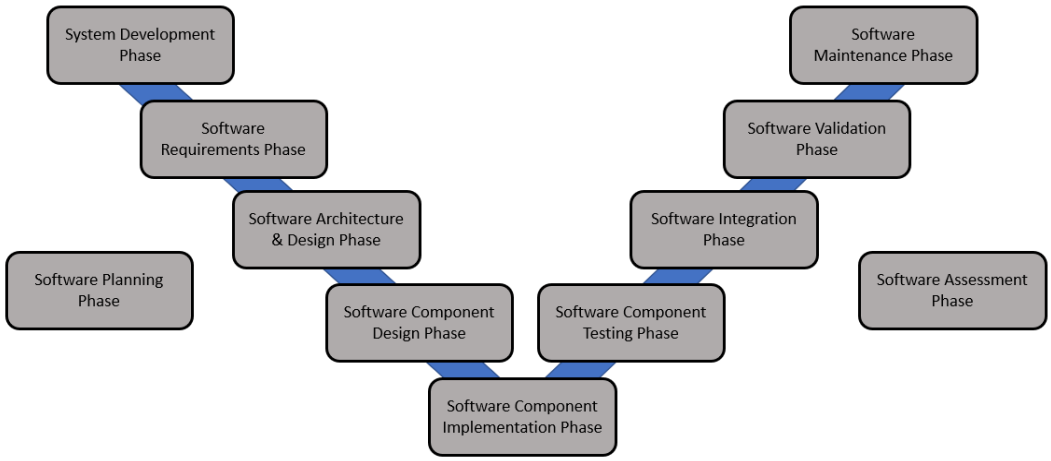

Figure. 1. IEC62279 Phases Overview - Adapted from [8]

2.4. IEC 62304

According to Magnusson [29], all medical devices must comply with regulations to ensure user and patient safety. With the increased use of software on medical devices, entities such as the US Food and Drugs Administration (FDA) have identified the need for specific software regulation. In 2006, a new international standard was released for Biomedical Software (BS) developed by a joint working group of the International Electrotechnical Commission (IEC). As a result, the use of IEC 62304 [9] has become fully harmonized in the United States and Europe.

IEC 62304 describes 5 processes: Software Development Process, Software Maintenance Process, Software Risk Management Process, Software Configuration Management Process, Software Problem Resolution Process, as shown in Figure 3. It requires manufacturers to assign a safety class for systems with software. This classification is based on the potential hazard that could result in injury to the user or patient in the event of abnormal system behaviour. The software can be categorized into three classes. Class C is of the utmost rigor and requires compliance with all associated activities. Classes B and A have a lower number of required

activities, as shown in Table 4. IEC 62304 has a large set of associated activities, described throughout the standard, relating to the development of Software Requirements. Regarding Software Requirements development, there is no difference between classes A, B, and C.

Among the activities associated with requirements development, the following stand out:

1. For each Software of a medical device, the manufacturer shall define and document the Software Requirements from the System Requirements (IEC 62304, Section 5.2.1); and

2. As appropriate to the Software of a medical device, the manufacturer shall include in the Software Requirements:

   a) The functional requirements, including performance, physical characteristics and computing environment (IEC 62304, Section 5.2.2 (a));

   b) Inputs and outputs, including data characteristics, value range, limits and typical values (IEC 62304, Section 5.2.2 (b)); and

   c) Software and system interfaces, including compatibility considerations (IEC 62304, Section 5.2.2 (c)).

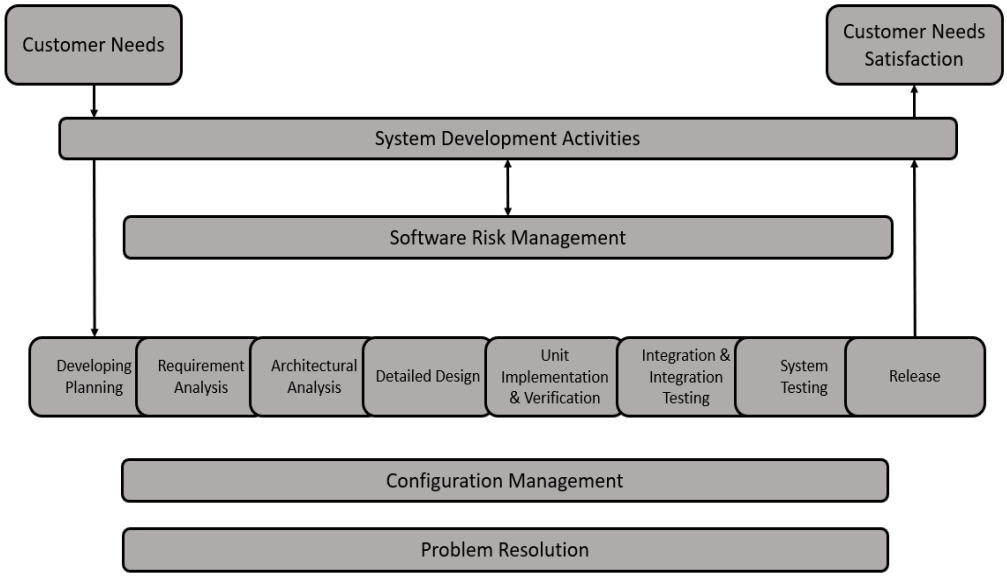

Figure. 2. IEC 62304 Process Overview - Adapted from [9]

Table 4. IEC 62304 software classes and total associated activities [30]

Class	Possible effects on the patient, operator, or other people	Associated Activities
A	No injury or damage to health is possible	44
B	Non-serious injury is possible	87
C	Death or serious injury is possible	92

During the review activity, for each defined software requirement, the manufacturer shall review, ensure and document that the Software Requirements:

1. Implement the defined System Requirements (IEC 62304, Section 5.2.6 (a));

2. Have no conflict with each other (IEC 62304, Section 5.2.6 (b));

3. They are correctly expressed, avoiding ambiguity (IEC 62304, Section 5.2.6 (c));

4. Have writing in such a way as to enable the establishment of a test criterion and to determine whether it has been met (IEC 62304, Section 5.2.6 (d));

5. Are uniquely identified (IEC 62304, Section 5.2.6 (e)); and

6. Provide traceability to system requirements (IEC 62304, Section 5.2.6 (f)).

The manufacturer must transform the Software Requirements of a medical device into a documented architecture that describes the Software structure by identifying the Software modules and components present (IEC 62304, Section 5.3.2). In addition, the manufacturer shall develop and document the architecture, including the interfaces between modules and external components (IEC 62304, Section 5.3.3).

3. ANALYSIS OF CHARACTERISTICS OF REQUIREMENTS ENGINEERING IN REGULATED ENVIRONMENTS

Typically, a good set of requirements needs minimal characteristics that allow it to be consistent, unambiguous, consistent, verifiable and traceable [31][32].

3.1. Internal and External Consistency

A set of Software Requirements is consistent if and only if all requirements expressed in any of the requirements do not conflict with the others. There are two types of consistency predicted in the literature [33][34][35][36]:

1. External Consistency refers to ensuring that Software Requirements are consistent with the inputs required for their formulation; and

2. Internal Consistency refers to ensuring that the Software Requirements are consistent with each other, ensuring that they are not inconsistent with each other.

RTCA DO-178C explicitly describes the internal consistency in section 6.3.1b (Objective A3-2) during the Software Requirements review process, where requirements must be evaluated to ensure that they do not conflict with each other. IEC 62304 deals with consistency in section 5.2.6b, where it is stated that the Software Requirements should be checked to ensure that they do not contradict each other. IEC 62279 stresses that internal consistency should be maintained in the revision of the Software Requirements set at the time of its formal revision, as explicitly stated in section 7.2.4.22e.

Regarding external consistency, RTCA DO-178C speaks of meeting System Requirements with Software Requirements in section 6.3.1a (Objective A3-1). IEC 62304 already states that Software Requirements shall implement the System Requirements in section IEC 62304, Section 5.2.6a. Finally, IEC 62279 also states in section 7.2.4.22a that the Software Requirements must comply with the System Requirements set. In a way, external consistency is understood in these

three standards, although not explicitly defined, as when deploying System Requirements, potential conflicts need to be assessed in this deployment.

3.2. Unambiguity

A set of Software Requirements is unambiguous if all requirements expressed in it have only one interpretation. At a minimum, this requires that each feature of the software product be described using simple and unique terms. Where a term used in each context may acquire multiple meanings, that term should be included in a glossary where its meaning is made more specific [33][34].

RTCA DO-178C addresses the ambiguity issue as part of accuracy and consistency, spelling out in section 6.3.1b that objective A3-2 should ensure that a Software Requirement is unambiguous. IEC 62304 provides in section 5.2.6c that the Software Requirements must be expressed in terms that avoid ambiguity. Finally, IEC 62279 provides in section 7.2.4.4 that the Software Requirements must be set unambiguously.

3.3. Verifiability

A set of Software Requirements is considered verifiable and testable if it can be verified that functional requirements and quality attributes have been properly implemented in design and code [31]. According to the International Standardization Organization [32], a requirement is verifiable if and only if there is a finite and acceptable cost process by which a person or machine can verify that the software product meets that requirement. In general, an ambiguous requirement is not verifiable.

In RTCA DO-178C, there is a need to prove that both Software High-Level Requirements (SW-HLR) and Software Low-Level Requirements (SW-LLR) are verifiable. This is defined in RTCA DO-178C Section 6.3.1 (d) for SW-HLR and RTCA DO-178C Section 6.3.2 (d) for SW-LLR.

In IEC 62304, there is also a need to ensure verifiability of Software Requirements. IEC 62304 Section B.5.2 describes that establishing verifiable requirements is essential in determining what should be built, and in determining whether medical device software behaves acceptably and is ready for use. To demonstrate that requirements have been implemented as desired, each requirement must be stated in such a way that objective criteria can be established to determine if it has been implemented correctly.

In IEC 62279, the verifiability of the requirements is also mandatory. However, it simply states in section 7.2.4.4 (a) that the software specification should be verifiably expressed without providing further details.

3.4. Traceability

According to Lauesen [21], requirements tracing is required to compare requirements with other related information. Requirement traceability is defined by Gotel & Finkelstein [36] as “the ability to describe from its origins, development and specification, to its subsequent deployment and use, and to periods of continuous refinement and iteration at any stage”.

In RTCA DO-178C, requirement traceability happens in many ways. Software High-Level Requirements (SW-HLR) shall have bidirectional traceability to System Requirements, Software Low-Level Requirements (SW-LLR) and test cases. Software Low-Level Requirements (SW-

LLR), which are part of Design, need to have bidirectional traceability for Software High-Level Requirements (SW-HLR), Source Code and Testing.

In IEC 62304, requirements traceability is less extensive than in RTCA DO-178C. Software Requirements must have bidirectional traceability to System Requirements and Testing. Although not mandatory, IEC 62304 provides that traceability between the Architecture, which is part of Design, and the Software Requirements may be useful for verification.

In IEC 62279, bidirectional requirements traceability happens from Software Requirements, to System Requirements, Design and Testing. Figure 4 presents a synthesis involving the traceability between Software Requirements and the other artefacts for the three software standards under analysis.

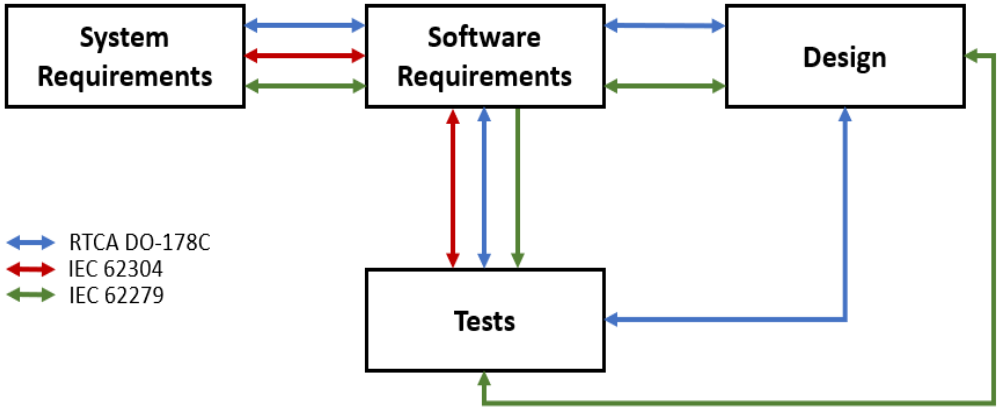

Figure. 3. Overview of Software Requirements Traceability

4. CONCLUSION

As presented in section 1, the objective of this paper was to present the software standards (RTCA DO-178C [7], IEC 62279 [8] and IEC 62304 [9]), addressing their similarities within the scope of requirements This work represents part of a research effort, conducted at the Aeronautics Institute of Technology, aiming at the possibility of proposing a universal framework that can simultaneously produce adherence to the three standards that are part of the scope of this work DO-178C [7], IEC 62279 [8] and IEC 62304 [9]. The framework to be designed should not only capture the Requirements Specification needs, but also the other features typically observed in Software Development processes in Regulated Environments, such as Architecture, Design, Code, Testing, among other aspects.

It is observed from the analysis conceived in section 3 that in the Requirements Specification scenario, the standards used have strong similarity characteristics. The following characteristics were analysed: consistency, unambiguity, verifiability and traceability. A cross-case analysis was performed according the instructions provided by de Jong [37]. Table 5 presents the cross-case analysis performed.

Table 5. Cross-case Analysis among Characteristics and Software Standards

Characteristic	Sub-characteristic	Cases
Consistency	Internal	RTCA DO-178C IEC 62279 IEC 62304
	External	RTCA DO-178C IEC 62279 IEC 62304
Unambiguity	None	RTCA DO-178C IEC 62279 IEC 62304
Verifiability	Test against requirements	RTCA DO-178C IEC 62279 IEC 62304
	Test against design	RTCA DO-178C IEC 62279
Traceability	Between system requirements and software requirements	RTCA DO-178C IEC 62279 IEC 62304
	Between software requirements to design	RTCA DO-178C IEC 62279
	Between software requirements to Tests	RTCA DO-178C IEC 62279 IEC 62304
	Between design to tests	RTCA DO-178C IEC 62279

The analysis of the first three characteristics (Consistency, Unambiguity and Verifiability) shows that standards require these characteristics to be verified. However, the fact that these three characteristics are provided for in these standards makes them mandatory from a regulatory standpoint, preventing companies in each domain from fulfilling their requirements specifications without considering them. The standards differ on Traceability, as presented in section 3.4.

This work represents an initial research effort aiming a future requirements framework universal process that can be configured to address each standard by the usage of Software Process Lines (SPL) that can simultaneously produce adherences to the three norms that are part of the scope of this work. The future framework will be composed for three different Software Process Lines (SPL) that can be configured for each domain (Aviation, Medical or Railway).

REFERENCES

[1] Marques, J., Cunha, A.M. (2015) *"Use of the RTCA DO-330 in Aeronautical Databases"*. In: Proceedings of the *34th IEEE/AIAA Digital Avionics System Conference*. United States.

[2] Marques, J., Cunha, A.M. (2018) *"Tailoring Traditional Software Life Cycles to Ensure Compliance of RTCA DO-178C and DO-331 with Model-Driven Design"*. In: Proceedings of the *37th IEEE/AIAA Digital Avionics System Conference*. United States.

[3] Fatwanto, A. (2012) *"Translating software requirements from natural language to formal specification"*. In: Proceedings of *the 2012 IEEE International Conference on Computational Intelligence and Cybernetics (CyberneticsCom)*. Indonesia.

[4] Ilyas, F., Zahra, K., Ambreen, N., Butt, W. (2016) *"A Survey on Current Requirement Process Practices in Software Companies and Requirement Process Problems"*. In: Proceedings of the *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*. United States.

[5] Sabrye, A.O.J., Zaainon, W.M.N.W. (2017) *"A framework for detecting ambiguity in software requirement specification"*. In: Proceedings of the *2017 8th International Conference on Information Technology (ICIT)*.

[6] Munch, J., Armbrunt, O., Kowalczyk, M., Soto, M. (2012) Software Process Definition and Management. 1st edn. Springer-Verlag, Germany.

[7] Radio Technical Commission for Aeronautics (2011) *DO-178C - Software Considerations in Airborne Systems and Equipment Certification*.

[8] International Electrotechnical Commission (2015) *IEC 62279:2015 Railway applications – Communications, signaling and processing systems – Software for railway control and protection systems*.

[9] International Electrotechnical Commission (2015) *IEC 62304:2015 Medical device software - Software life-cycle processes.*

[10] Marques, J., Cunha, A.M. (2013) *"A Reference Method for Airborne Software Requirements"*. In: Proceedings of the *32nd IEEE/AIAA Digital Avionics System Conference*. United States.

[11] Marques, J., Cunha, A.M. (2017) Especificação de Requisitos de Software: Um Modelo Ágil para Ambientes Regulados. 1 st edn. Novas Edições Acadêmicas, Brazil.

[12] Radio Technical Commission for Aeronautics (2011) *DO-278A - Software Integrity Assurance Considerations for Communication, Navigation, Surveillance and Air Traffic Management (CNS/ATM) Systems*.

[13] International Standardization Organization (2011) *ISO26262-6 Road vehicles – Functional safety – Part 6: Product development at the software level.*

[14] European Cooperation for Safety Standardization (2009) *ECSS-E-ST-40C Space Engineering – Software*.

[15] International Atomic Energy Agency (2016) *Specific Safety Guidance SSG-39: Design of Instrumentation and Control Systems for Nuclear Power Plants*.

[16] Marques, J., Cunha, A.M. (2017) *"Verification Scenarios of Onboard Databases under the RTCA DO-178C and the RTCA DO-200B"*. In: Proceedings of the *36th IEEE/AIAA Digital Avionics System Conference*.

[17] Federal Aviation Administration (2018) *AC20-115D Airborne Software Development Assurance Using EUROCAE ED-12( ) and RTCA DO-178( ).*

[18] Society of Automotive Engineers (2010) *ARP 4754 - Guidelines For Development Of Civil Aircraft and Systems.*

[19] Yelisetty, S.M.H., Marques, J., Tasinaffo, P. (2015) *"A Set of Metrics to Assess and Monitor Compliance with RTCA DO-178C"*. In: Proceedings of the *34th IEEE/AIAA Digital Avionics System Conference*.

[20] Radio Technical Commission for Aeronautics (2011) *DO-330 - Software Tool Qualification and Considerations*.

[21] Radio Technical Commission for Aeronautics (2011) *DO-331 - Model-Based Development and Verification Supplement to DO-178C and DO-278A*.

[22] Radio Technical Commission for Aeronautics (2011) *DO-332 - Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A*.

[23] Radio Technical Commission for Aeronautics (2011) *DO-333 - Formal Methods Supplement to DO-178C and DO-278A*.

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

[25] European Aviation Safety Agency (2011) *Certification Memo SW-CEH 002*. Germany.

[26] Joung, E. J., Lee C. M., Lee H. M., Kim G. D. (2009) "Software Safety Criteria and Application Procedure for the Safety Critical Railway System". In: Proceedings of the *2009 Transmission & Distribution Conference & Exposition: Asia and Pacific*, pp: 1-4. South Korea.

[27] Chen, T. (2017) "Non-safety-related software in the context of railway RAMS standards". In: Proceedings of the Second International Conference on Reliability Systems Engineering (ICRSE), pp: 1-5. China.

[28] Wigger, T. (2001) *"Experience with Safety Integrity Level (SIL) Allocation in Railway Applications"*. In: Proceedings of the *World Congress Railway Research*.

[29] Magnuson, A. (2012) IEC/ISO 62304 Regulations for the Development of Medical Software Devices. Master's Thesis Chalmers University of Technology. Sweden.

[30] Marques, J., Cunha, A (2019) ARES: An Agile Requirements Specification Process for Regulated Environments. In: International Journal of Software Engineering and Knowledge Engineering Vol. 29, No. 10, World Scientific Publishing Company, pp: 1403-1438.

[31] Institute of Electrical and Electronics Engineers (1998) *IEEE 830-1998 - IEEE Recommended Practice for Software Requirements Specifications.*

[32] International Standardization Organization (2011) *ISO/IEC/IEEE 29148:2011 ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes --Requirements engineering*.

[33] Dick, J., Hull, E., Jackson, K. (2017) Requirements Engineering. 4th edn. Springer. United States.

[34] Laplante, P. (2013) Requirements Engineering for Software and Systems, 3rd edn. CRC Press. United States.

[35] Lauesen, S. (2002) Software Requirements – Styles and Techniques. 1st edn. Pearson. United Kingdom.

[36] Gotel, O. C. Z., Finkelstein, C. W. (1994) *"An analysis of requirements traceability problem"*. In: Proceedings of the *IEEE International Conference on Requirements Engineering*. United States.

[37] de Jong T. (2010) *"Cross case analysis: Relating the empirical findings"*. In: Linking social capital to knowledge productivity. Germany.

AUTHORS

Johnny Marques Was Born In Toronto, Canada, In 1977, But Has Been Living In Brazil Since 1986. He Received The B.Sc. In Computer Engineering From University Of The State Of Rio De Janeiro (UERJ), The M.Sc. (In Aeronautical Engineering) And A Phd. (In Electronic And Computer Engineering) Both From Aeronautics Institute Of Technology (ITA). He Is A Current A Professor In The Aeronautics Institute Of Technology (ITA). He Worked At EMBRAER In Software Processes Definition For 15 Years And Has Experience In Standards Used For Airborne Systems And Software Such As DO-178C, DO-254, ARP-4754, And DO-200B. He Is Also Part Of Several Committees In IEEE Standards Association.

Sarasuaty Yelisetty Was Born In Sao Jose Dos Campos, Sao Paulo, Brazil. She Received The B.Sc. In Computer Engineering From University Of Vale Do Paraiba (UNIVAP). She Received The M.Sc. In Computer And Electronic Engineering From Aeronautics Institute Of Technology (ITA). Currently, She Is A Phd Student At ITA. Additionally, She Has Been Working At EMBRAER During The Last 10 Years And Has Experience In Standards Used For Airborne System And Software Certification.