# Qualification Considerations for Simulations in Avionics Software Engineering

Mohamad Ibrahim[1*], Umut Durak[2]

[1]Institute of Informatics, TU Clausthal, Julius-Albert-Str. 4,
38678 Clausthal-Zellerfeld, Germany; *mohamad.ibrahim@tu-clausthal.de

[2]Instutute of Flight Systems, German Aerospace Center (DLR), Lilienthalplatz 7, 38108 Braunschweig, Germany

**Abstract.** Simulation engineering has become an established practice in software development processes in various domains. ARP4754A Guidelines for Development of Civil Aircraft and Systems and DO-178C Software Considerations in Airborne Systems and Equipment Certification have adopted simulation to aid in requirements validation and implementation verification. Tool qualification requirements have already been settled in safety-critical domains. However, the methods and guidelines for applying these requirements in the simulation engineering life cycle are still missing. This work introduces the background in tool qualification, presents the state of the art, and discusses the qualification considerations for simulation in avionics software engineering.

## Introduction

Simulation engineering is performed in the process of software development for different domains. It is a method that supports the verification and validation of software systems and helps to uncover and assess risk factors. Simulation engineering is the execution of an interdisciplinary systems engineering process for developing, maintaining, and employing simulations, which enable systems engineers to experiment and gain insight into the systems of interest [1].

Simulation has been widely adopted in safety-critical software development. ARP4754A Guidelines for Development of Civil Aircraft and Systems referred to the use of simulation to achieve requirements validation and implementation verification. Requirements validation ensures correctness and completeness to meet the stakeholders' needs such as flight crew or developers. While implementation verification ensures the conformance of the system implementation to its validated requirements.

In the avionic domain, developers use simulation with different software artifacts, namely, Specification Models, Design Models, and executable code. DO-331 [2], the model-based development and verification supplement to DO-178C promotes simulation as a way to satisfy its objectives [3]. DO-331 accepts simulation as means to satisfy the verification objectives of the specification and design models, however, it accepts simulation only in combination with testing and coverage analysis to satisfy the verification objectives of the executable code.

This practice of using simulation to satisfy certification or development assurance objectives makes the correctness of simulation a safety concern. Besides, as the complexity of the system increases, its simulation also becomes remarkably complex, that it is necessary to assess it, not only as a valid support for systems engineering processes but also as an objective of systems engineering efforts. One approach to address this safety concern is through the adoption of tool qualification as a framework for quality assurance.

Tool qualification is "a process that allows us to demonstrate that a tool can be used as part of the realization of a software application with a determined safety goal"[4]. Tool qualification guarantees that the tool is developed and verified using an adequate process to obtain confidence in the tool's functionality.

IEEE 1730 Distributed Simulation Engineering and Execution Process (DSEEP) is a recommended practice for simulation life-cycle processes. In conjugation with DO-330 [5], the Software Tool Qualification Considerations supplement to the avionics standards RTCA DO-178C, DSEEP can form a basis for a simulation qualification strategy.

**Structure of the paper**. Section 1 is a summarization of the most important aspects of tool qualification and then focuses on the DO-330 guidelines. In section 2, we discuss the current utilization of simulation engineering in the avionic system and software development. Section 3 is dedicated to paving the floor to the proposed approach. Finally, we conclude in section 4.

# 1 Tool Qualification

Tool qualification is the process of documenting pieces of evidence that show the tool is reliable and fit the intended purpose in the context of a specific use case. Boulanger defines tool qualification in [4] as “a process that allows us to demonstrate that a tool can be used as part of the realization of a software application with a determined safety goal”.

Software tools reside in two categories [6], Software Development Tools and Software Verification Tools. The categories contrast in which the development tools can insert an error in the software while verification tools can only fail to detect an error in the software [6]. Examples of Development Tools are compilers, linkers, modeling tools, code generators, code manipulators, etc. Verification Tools include test case generators, code static analysis, test automation, structural coverage tools, test results checker, etc.

All safety standards adopt the principle that says: “qualification of the tool is required only when this tool replaces, reduces, or automates one of the software lifecycle processes”. Nevertheless, the qualification process can be eliminated in case the output of the tool is verified by another process or a qualified tool. In other words, if the activities or tasks required by a standard rely on the correct functioning of the tool, then tool qualification is a necessity [7].

In general, there are four methods/approaches for tool qualification that is accepted by most standards [8]:

1. Increased confidence from use (proven in use argumentation)

2. Evaluation of the development process (process assessment)

3. Validation of the software tool in the operational environment

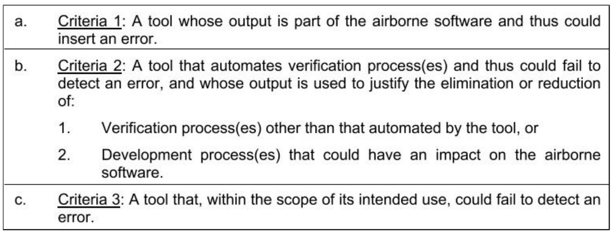
4. Development in compliance with a safety standard

The avionic standard for functional safety and certification aspects DO-178C and its tool qualification supplement DO-330 only accept the last two methods. The 3rd method represents TQL-5 (the lowest Qualification Level of DO-330), and the fourth method consists of presenting evidence of developing the tool according to a safety standard such as DO-178C, and thus it involves input from the developer. The rest of this section will focus only on the qualification process of the DO-178C/DO-330. This brief introduction to DO-330 is adopted from our previous work [9].

## 1.1 Tool Classification and Analysis

All qualification methods start with evaluating the tool’s impact on the process workflow of the software lifecycle. DO-178C defines three criteria (Tool Impact) for the Tool under Qualification:

**Table 1:** DO-178C Tool Impact Criteria [5]



| | |
|---|---|
| a. | Criteria 1: A tool whose output is part of the airborne software and thus could insert an error. |
| b. | Criteria 2: A tool that automates verification process(es) and thus could fail to detect an error, and whose output is used to justify the elimination or reduction of:<br>1. Verification process(es) other than that automated by the tool, or<br>2. Development process(es) that could have an impact on the airborne software. |
| c. | Criteria 3: A tool that, within the scope of its intended use, could fail to detect an error. |

Criteria 1 corresponds to the Development Tools category, while Criteria 2 and 3 represents verification tools. Criteria 3 tools should be extended to Criteria 2 if they are used for eliminating or reducing a process that is mandated to be used by DO-178C in the software life cycle.

Based on the criteria of the tool, finding the TQL is a matter of table lookup activity. The following table determines the TQL

**Table 2:** Tool Qualification Level Determination

| **Software Level** | **Criteria** | | |
|---|---|---|---|
| | ***1*** | ***2*** | ***3*** |
| A | TQL-1 | TQL-4 | TQL-5 |
| B | TQL-2 | TQL-4 | TQL-5 |
| C | TQL-3 | TQL-5 | TQL-5 |
| D | TQL-4 | TQL-5 | TQL-5 |

Based on the resulted target TQL, DO-330 specifies the set of objectives that the qualifier should fulfill to achieve the qualification of the tool. These objectives are set in processes that constitute the life cycle of the tool qualification.

## 1.2 Qualification Life Cycle and Processes

Software quality cannot be added to a product after it is developed [6]. Thus, DO-330 defines tool qualification life cycle processes (Fig.1 adopted from [10]) to qualify the tools and meet the required quality and assurance:

1. Tool Operational Process
2. Tool Planning Process
3. Tool Development Process
4. Tool Verification Process
5. Integral Processes is done throughout the entire tool qualification life cycle.

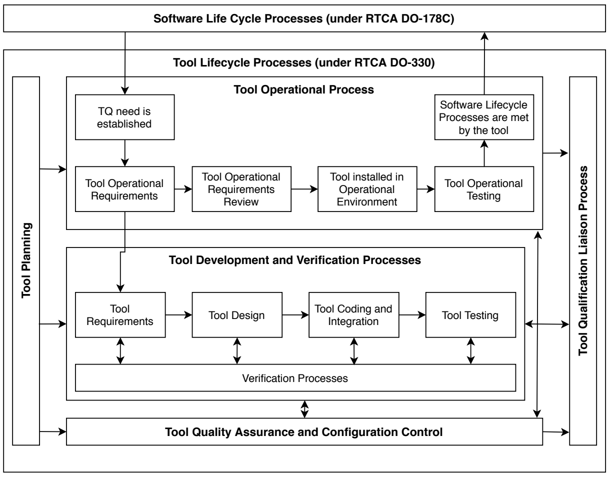

Software Life Cycle Processes (under RTCA DO-178C)
Tool Lifecycle Processes (under RTCA DO-330)
Tool Operational Process
TQ need is established
Software Lifecycle Processes are met by the tool
Tool Operational Requirements
Tool Operational Requirements Review
Tool installed in Operational Environment
Tool Operational Testing
Tool Planning
Tool Development and Verification Processes
Tool Requirements
Tool Design
Tool Coding and Integration
Tool Testing
Verification Processes
Tool Quality Assurance and Configuration Control
Tool Qualification Liaison Process

**Figure 1:** Tool Qualification Life Cycle

**Tool Operational Process** is the user's responsibility. The main objectives involved in this process include: tool qualification need is established, TOR (Tool Operational Requirements), tool executable code installed, and TOR is validated [10].

**Tool Planning Process** defines the tool qualification processes and their interaction/interrelationship for the tool qualification life cycle. In the planning process, standards, verification environment, and development environment should be defined. Lastly, the qualifier should enumerate the DO-330 objectives and describe how they will be fulfilled.

**Tool Development Process** is the implementation process of the tool. High-Level Requirements (HLR), Low-Level Requirements (LLR), and code are developed considering traceability, transition, and integration criteria in the process.

**Tool Verification Processes** is done sequentially in two phases, the first is the verification of the tool functional requirements in which the tool developers' work needs to be verified against the intended functionality (Fig. 1 the "Tool Development and Verification Processes" block). Secondly, is the tool operational verification and validation process in which the tool user's work needs to be verified against the intended usage (Fig. 1 the "Tool Operational Process" block).

## 2 Safety-critical Simulation Engineering in Airborne Development

Guidance for the development of airborne systems for civil aircraft can be found in ARP4754A [11]. It defines the development process as a set of activities from the early stages of conceptualization to certification final phases. ARP4754A emphasizes the simulation sufficiency, appropriateness, and validation. Notwithstanding, there is no clear definition of the simulation process nor its quality assurance requirements and objectives. DO-178C supports ARP4754A by providing process requirements for the entire software development life cycle. DO-178C and its supplements allude to simulation in two contexts:

1. Model Simulation Context: refers to the execution of the Specification or Design Model to collect pieces of evidence about the compliance of Specification Models to system requirements, compliance of Design Models to software high-level requirements, and further model qualities such as accuracy or consistency.
2. Executable Simulation context: it aims to reveal errors that usually arise from running the software on the target hardware. Such use mandates to prove that the simulator is representative of the actual target hardware.

The utilization of simulation requires a simulation tool or simulator. DO-178C requires in some cases the simulator to be qualified based on its use. The qualification process is well-defined in the DO-178C supplement DO-330. Thus, our work focuses on the quality of the simulation process itself and not on the quality of the simulation tool or simulator. Furthermore, for the sack of showcasing, we will be proposing an approach that addresses the model simulation context, in which the objectives are confined to the verification and validation of the design models.

The problem of verifying the correctness of the simulation models has long been one of the major concerns of the simulation engineering community in model-based software development. it dates back to the 1960s. A survey that addresses the work on this topic since the early days is made by Sargent et al. [12]. The subsequent research and academic efforts have led to the standardization of the modeling and simulation life cycle as part of the High-Level Architecture (HLA) standard suite. IEEE Std 1516.3-2003, IEEE Recommended Practice for High-Level Architecture (HLA) Federation Development and Execution Process (FEDEP), proposed a process for federation development, particularly for distributed simulations that utilize HLA [13].

FEDEP was then generalized by the Simulation Interoperability Standards Organization (SISO) FEDEP Product Development Group (PDG) to support the engineering process for all types of distributed simulation. This was published in 2010 as IEEE Std 1730-2010: IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP) [14]. With DSEEP we have got a strong basis to build our approach for the simulation qualification since it is currently the de facto standard and the most well-received standard process for simulation engineering.

DSEEP is a generalized process for simulation engineering [14]. Although it refers to distributed simulation engineering, it is also applicable for stand-alone applications. It presents the best practices through defining the processes and procedures to be followed to develop and execute simulations, following is the flow of the DSEEP processes:

- *Step 1: Define Simulation Environment Objectives* aims at specifying the needs and eventually objectives that are to be addressed with the simulation to be developed and executed.

- *Step 2: Perform Conceptual Analysis* aims at modeling, scenario development, and requirements specification.

- *Step 3: Design Simulation Environment* includes identifying member applications, reuse candidates, and planning.

- *Step 4: Develop Simulation Environment* includes developing data exchange models, simulation environment agreements, member application development, and infrastructure implementation.

- *Step 5: Integrate and Test Simulation Environment* aims at integrating all the member applications using the implemented infrastructure and testing the environment before execution.

- *Step 6: Execute Simulation* is the step where the simulation (all member applications) is executed and the results are collected.

- *Step 7: Analyze Data and Evaluate Results* aims at the analysis and the evaluation of the collected data

Similar to DO-330, each step in DSEEP is further branched to activities. Furthermore, each activity is concertized to activity input, activity output, and recommended tasks.

## 3 Simulation Qualification

This paper present an extension to the previous papers [15, 16, 17]. In this paper, the authors try to promote and plant the seeds for new practices that establish Simulation Qualification as a requirement in the avionic software development domain. Thus, we propose the utilization of DO-330 and DSEEP as the foundations for simulation qualification. The approach takes guidance from the qualification approach in the airborne system's domain -specifically DO-330- to achieve objectives mandated by DSEEP. This will facilitate its usage among DO-178C practitioners and experts and make it more acceptable and complied with the industry's high standards.

We propose not to update, enhance or push DSEEP more towards DO-330 [17, 16], but rather get DSEEP as a self-contained and complete simulation engineering life cycle process and trace its recommendations to the requirements for the qualification of simulations for safety-critical systems engineering [15].

Despite the structure of DO-330 and DSEEP are different, Both guidelines will fill in different parts of the proposed approach. E.g. the integral processes like verification and validation or configuration management are not explicit, but referred to and explained within the steps of DSEEP.

The approach initiates similar to DO-330, by evaluating the simulation impact as follow:

a Criteria 1: A simulation whose output is part of the system and thus can introduce an error

b Criteria 2: A simulation that is used in verification and validation of the system and this could fail to detect an error, and whose output is used to justify the elimination or reduction of another validation and verification effort.

c Criteria 3: A simulation, within the scope of its intended use, could fail to detect an error.

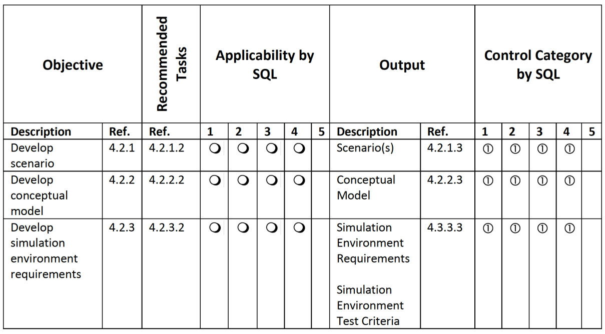
Subsequently, we build on the elected criteria to determine the SQL (Simulation Qualification Level), the lookup table is depicted in Table 3:

**Table 3:** Determination of SQL.

| **Criteria** | **Development Assurance Level** | | | |
|---|---|---|---|---|
| | **A** | **B** | **C** | **D** |
| **1** | SQL1 | SQL2 | SQL3 | SQL4 |
| **2** | SQL4 | SQL4 | SQL5 | SQL5 |
| **3** | SQL5 | SQL5 | SQL5 | SQL5 |

DO-330 describes the objectives as requirements that need to be fulfilled to demonstrate compliance with the document. We propose to adopt the DSEEP activities as the major objectives and enhance them with the inline statements in activity definitions regarding the support processes, or integral processes, such as verification and validation. As a sample we make an overlay for the DSEEP Perform conceptual analysis process, Table 4 depicts the objectives for this process.

**Table 4:** Perform Conceptual Analysis Step Objectives.

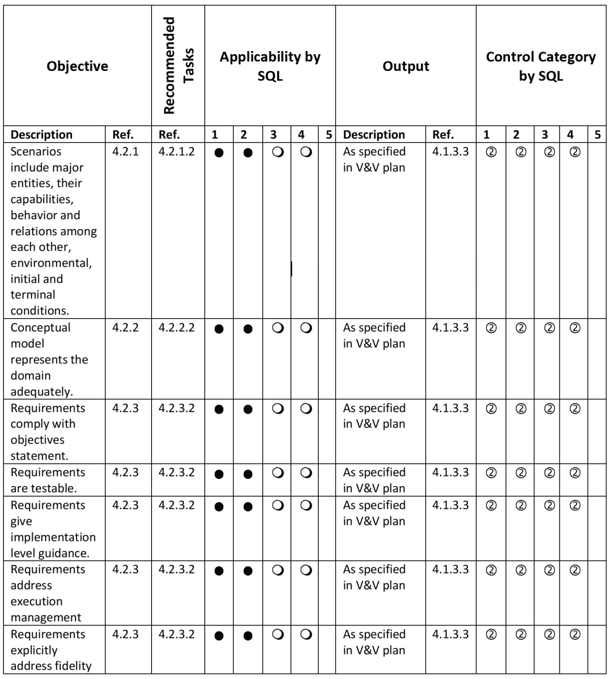


| **Objective** | | **Recommended Tasks** | **Applicability by SQL** | | | | | **Output** | | **Control Category by SQL** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Description** | **Ref.** | **Ref.** | **1** | **2** | **3** | **4** | **5** | **Description** | **Ref.** | **1** | **2** | **3** | **4** | **5** |
| Develop scenario | 4.2.1 | 4.2.1.2 | ❍ | ❍ | ❍ | ❍ | | Scenario(s) | 4.2.1.3 | ① | ① | ① | ① | |
| Develop conceptual model | 4.2.2 | 4.2.2.2 | ❍ | ❍ | ❍ | ❍ | | Conceptual Model | 4.2.2.3 | ① | ① | ① | ① | |
| Develop simulation environment requirements | 4.2.3 | 4.2.3.2 | ❍ | ❍ | ❍ | ❍ | | Simulation Environment Requirements<br>Simulation Environment Test Criteria | 4.3.3.3 | ① | ① | ① | ① | |

The objectives are listed as: Develop scenario, Develop conceptual model, and Develop simulation environment requirements. The reference for the objective points to the related section in DSEEP that explains the activity. The recommended Tasks replace the Activity references in DO-330; they are the means to meet the objectives. Aligned with the requirements engineering objectives of DO-330, the objectives apply to all SQLs from 1 to 4. The Outputs refer to DSEEP activity outputs. For example, as described in section 4.3.3.3 of DSEEP, Simulation Environment Requirements and Simulation Environment Test Criteria are the outputs of the Develop Simulation Environment objective. The control categories for all objectives are assigned as 1.

The objective tables of DO-330 are organized in such a way that there is a table for the verification of outputs of each development process. Accordingly, Table 5 lists the objectives for the verification of the outputs of Perform Conceptual Analysis step.

**Table 5:** Verification of Conceptual Analysis.



| **Objective** | | **Recommended Tasks** | **Applicability by SQL** | | | | | **Output** | | **Control Category by SQL** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Description** | **Ref.** | **Ref.** | **1** | **2** | **3** | **4** | **5** | **Description** | **Ref.** | **1** | **2** | **3** | **4** | **5** |
| Scenarios include major entities, their capabilities, behavior and relations among each other, environmental, initial and terminal conditions. | 4.2.1 | 4.2.1.2 | ● | ● | ❍ | ❍ | | As specified in V&V plan | 4.1.3.3 | ② | ② | ② | ② | |
| Conceptual model represents the domain adequately. | 4.2.2 | 4.2.2.2 | ● | ● | ❍ | ❍ | | As specified in V&V plan | 4.1.3.3 | ② | ② | ② | ② | |
| Requirements comply with objectives statement. | 4.2.3 | 4.2.3.2 | ● | ● | ❍ | ❍ | | As specified in V&V plan | 4.1.3.3 | ② | ② | ② | ② | |
| Requirements are testable. | 4.2.3 | 4.2.3.2 | ● | ● | ❍ | ❍ | | As specified in V&V plan | 4.1.3.3 | ② | ② | ② | ② | |
| Requirements give implementation level guidance. | 4.2.3 | 4.2.3.2 | ● | ● | ❍ | ❍ | | As specified in V&V plan | 4.1.3.3 | ② | ② | ② | ② | |
| Requirements address execution management | 4.2.3 | 4.2.3.2 | ● | ● | ❍ | ❍ | | As specified in V&V plan | 4.1.3.3 | ② | ② | ② | ② | |
| Requirements explicitly address fidelity | 4.2.3 | 4.2.3.2 | ● | ● | ❍ | ❍ | | As specified in V&V plan | 4.1.3.3 | ② | ② | ② | ② | |

The objectives are based on the text that describes the corresponding activity in DSEEP. Simply explained, "shall" statements are reworded as verification objectives. An example would be "Requirements comply with the objectives statement". The objective and the recommended tasks can be found in sections 4.2.3 and 4.2.3.2

of DSEEP, respectively. The objective shall be fulfilled for the SQLs 1 to 4, and with independence, designated by the solid dot, for the SQL-1 and SQL-2.

## 4 Conclusion and Outlook

The paper presents an overview of the Tool Qualification process and a brief overview of the simulation engineering processes in the avionic domain. Upon that, it constructs an approach to establish a framework for the qualification of simulations to be used in safety-critical systems engineering. The proposed framework extends DO-330 with simulation processes and objectives from DSEEP to achieve quality assurance in the model simulation process. It is important to note that the paper relies on DSEEP as a self-contained and complete process framework for simulation engineering. It proposes to augment it with definitions of Simulation Qualification Levels (SQLs) and a set of objectives that are required to achieve these SQLs. This approach forms a firm ground for future research in the quality assurance of the simulation process under safety-critical software development. The further research aims to extend best practices in simulation engineering in a widely accepted guideline document that makes simulation a more reliable means of standard compliance.

### References

[1] Durak U, Ören T. Towards an ontology for simulation systems engineering. In: *Proceedings of the 49th Annual Simulation Symposium*, ANSS '16. Pasadena, California: Society for Computer Simulation International. 2016; pp. 1–8.

[2] RTCA. DO-331 The model-based development and verification supplement to DO-178C. *Radio Technical Commission for Aeronautics*. 2012;.

[3] RTCA. DO-178C Software Considerations in Airborne Systems and Equipment Certification. *Radio Technical Commission for Aeronautics*. 2011;.

[4] Boulanger JL. *Certifiable Software Applications 2: Support Processes*. Elsevier. 2017.

[5] RTCA. DO-330 Software Tool Qualification and Considerations. *Radio Technical Commission for Aeronautics*. 2011;.

[6] Hilderman V. DO-330: Tool Qualification Overview for Avionics Engineers and Managers. *wwwafuzioncom*. 2017;.

[7] Conrad M, Sandmann G, Munier P. Software Tool Qualification According to ISO 26262. In: *SAE 2011 World Congress Exhibition*. SAE International. 2011; .

[8] ISO. 26262: Road vehicles-Functional safety. *International Standard ISO/FDIS*. 2011;26262.

[9] Ibrahim M, Durak U. State of the Art in Software Tool Qualification with DO-330: A Survey. In: *Proceedings of the Software Engineering 2021 Satellite Events, Braunschweig/Virtual, Germany, February 22 - 26, 2021*, edited by Götz S, Linsbauer L, Schaefer I, Wortmann A, vol. 2814 of *CEUR Workshop Proceedings*. CEUR-WS.org. 2021; .

[10] Marques J, Marques da Cunha A. COTS tool qualification using RTCA DO-330: Common pitfalls. In: *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*. 2017; pp. 1–6.

[11] SAE. ARP4754A Guidelines for Development of Civil Aircraft and Systems. *SAE International*. 2010;.

[12] Sargent RG, Balci O. History of verification and validation of simulation models. In: *2017 Winter Simulation Conference (WSC)*. IEEE. 2017; pp. 292–307.

[13] IEEE. IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP). *IEEE Std 15163-2003*. 2003;.

[14] IEEE. IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP). *IEEE Std 1730-2010*. 2011;.

[15] Durak U, D'Ambrogio A, Bocciarelli P. Safety-Critical Simulation Engineering. In: *Proceedings of the 2020 Summer Simulation Conference*, SummerSim '20. San Diego, CA, USA: Society for Computer Simulation International. 2020; .

[16] Durak U, D'Ambrogio A, Gerlach T. Applying IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process for Mod-

eling and Simulation Based Airborne Systems Engineering. In: *AIAA Scitech 2020 Forum*. 2020; p. 0896.

[17] Mahmoodi S, Durak U, Hartmann S, Jafer S. DO-330/ED-215 Overlay to the IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process. *Journal of Aerospace Information Systems*. 2018;15(12):696–705.