IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 25, 2020, accepted July 25, 2020, date of publication August 14, 2020, date of current version August 24, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3014794*

# A Framework for Evaluating the Standards for the Production of Airborne and Ground Traffic Management Software

**JOSE ANDRES-JIMENEZ, JOSE-AMELIO MEDINA-MERODIO, LUIS FERNANDEZ-SANZ, JOSE-JAVIER MARTINEZ-HERRAIZ, AND JAVIER GONZALEZ-DE-LOPE**

Department of Computer Science, Polytechnic School, University of Alcalá, 28871 Alcalá de Henares, Spain

Corresponding author: Jose Andres-Jimenez (jose.andresj@edu.uah.es)

**ABSTRACT** The development of Airborne and Ground systems is framed by specific regulations, usually expressed as standards. A disadvantage of those standards is the inherent complexity for its application and the verification of compliance given the high number of requirements to be checked in many different situations of the application and which are highly dependent on the applicable level of criticality. When the development of this type of system requires the incorporation of new personnel without enough knowledge about the standards, risks of mistakes in their application grow exponentially. The objective of this work is to develop an Expert System (ES) that helps to evaluate the application of the standards DO-178C and DO-278A throughout the project life cycle, at the same time it serves to facilitate both its use and the learning of its application to a wide group of professionals. The proposed underlying method for the ES will allow evaluating the set of development processes to check coverage of the standards DO-178C and DO-278A without depending on a specific life cycle model. The method involves a model of the set of processes, so they can be evaluated by the ES. Additionally, the ES will require a minimum configuration to evaluate the development of systems based on these two standards. The main result is a new generic Expert System based on rules capable of being adapted to different environment of evaluation, whichh minor configuration operations thus allowing that a Generic ES can act as a Specific ES for each situation. This configurable ES has been customized to evaluate the software life cycle based on the standards under study.

**INDEX TERMS** Rule-based expert systems, process management, inference engine, knowledge base, artificial intelligence, JRuleEngine, checklists, software life cycle, DO-178C, DO-278A, safety critical systems.

## I. INTRODUCTION

The design and development of Airborne and Ground Systems requires the use of expert knowledge in several disciplines, and software engineering is one of them. This field is highly conditioned by the existence of complex standards, which require extensive knowledge to ensure the compliance of systems with them.

As many activity areas (such as quality assurance, development, validation, verification, configuration management, etc.) are involved in those applicable standards, reaching and showing the overall compliance is even harder than in other projects. The more complex is the system, the more difficult is the development and the more demanding is the subsequent verification. Therefore, it is necessary to provide new techniques to improve the ability of engineers to develop, verify and validate these systems in the usual conditions of limited time and resources.

Airborne and ground systems are closely related to critical systems where failure may lead to severe risks like injury or loss of life [1]: in this context it also involves the loss of the aircraft. Airborne systems may support a large variety of functions, ranging from those directly impacting safety in flight to functions with a minor effect or no effects on safety. Obviously, the systems with significant impact on safety are the most demanding in Planning, Development,

The associate editor coordinating the review of this manuscript and approving it for publication was Biju Issac.

Configuration Management, Quality Assurance, and, of course, Verification. Their development is embedded within specific regulations defining the guidelines and requirements for design, implementation, and validation. At the same time, this type of systems requires the intervention and cooperation of different and varied areas of activity, rules or specific methods not only focused on software development but also on hardware development, configuration, integrated systems and safety. Additionally, systems such as, e.g., Air Traffic Management Systems which are ground systems, share similar requirements and regulations as the airborne systems.

The Radio Technical Commission for Aeronautics (RTCA) has published DO-178C [2] and DO-278A [3] to provide guidance for the production of airborne and ground traffic management software. They are respectively being used in the Certification and Approval of Safety-Critical Software [4]. Standards should be supported by strong verification processes along the life cycle. Some studies [5], [6] show that the use of checklists as a formal review technique really helps to the evaluation of critical systems. However, the review techniques may represent a large expenditure of qualified resources to ensure compliance with standards. One of the objectives of our work is supporting these techniques through their integration within intelligent systems, which may help not only in implementing them but also in verifying compliance with them. The checklist will be the pivotal item for the proposed system.

Integration of Artificial Intelligence (AI) in the development of Critical Systems is not new as one can have already seen in different areas of applications: Aviation, Automotive, Medicine, Nuclear and Control Systems, among others. Its implementation relies on a wide variety of techniques such as Expert Systems, Fuzzy Logic or Neural Networks. In fact, it has been used to improve the design process [7], reducing development time and costs. But our vision is also adding AI to the assessment of standard compliance, while also serving as a learning guide for software professionals involved in the development of critical systems. As rule-based Systems are an efficient tool to address complex situations governed by deterministic rules such as traffic control systems, safety systems, bank transactions, etc. [8], this type of systems allows capturing the human experience in solving problems, in order to reach consistent and repeatable decisions [9]. This is the reason why we selected this type of solution in the form of Expert System (ES) for our study.

The objective of this work is to develop an Expert System to evaluate the application standards DO-178C [2] and DO-278A [3] along the life cycle and thus facilitating the use of standards, while also helping professionals to learn more on their characteristics and implications.

This article is organized as follows. Section II presents a short review of the techniques and standards, which will act as reference. Section III highlights the research methodology while section IV presents the practical development of the proposal. Section V analyses and discusses the results collected from practice and finally, Section VI presents conclusions and section VIII the future works.

## II. RELATED WORK

### A. DEVELOPMENT OF CRITICAL SYSTEMS

Critical Systems are present in several application environments: Transportation (Aviation, Automotive, Railway), Aerospace, Medical, and Industrial (Nuclear Plants) Sectors. Three types of components can be the origin of a system failure in this type of systems [1]: hardware, software and operators and users of the system who may cause failures through different actions. Our proposal of ES is aimed at reducing the probability of errors related to software development in the Aviation sector.

One of the key points when developing a critical system is the choice of a life cycle model. There are different models namely ranging from the most traditional such as Waterfall model, Incremental model, V-model, Spiral model or prototyping to the boom of Agile Methodologies at present times [10] [11]. Each model has advantages and disadvantages [12].

DO-178C [2] and DO-278A [3] do not define a particular life cycle, so the best option for our intended ES is making it work as a tool to help determine whether a particular process meets a set of objectives and activities regardless of the used life cycle. As the development of a critical system may involve many different disciplines and different application domains, we can find many specific standards for each case; Development (ARP-4754A [13]), Safety (ARP-4761 [14], ED-153 [15]), Processes (IEC-61508 [16]), Hardware (DO-254 [17]), Software (DO-178C [18], DO-278A [3]), Automotive (ISO-26262 [18]), Nuclear (IEC-61513 [19], IEC-61226 [20]), Railway (EN-50126 [21], EN-50128 [22], EN-50129 [23]), Medical (IEC-60601 [24], IEC-62304 [25]), etc.

All these standards work under schemes of criticality levels such as the Development Assurance Level (DAL), Safety Integrity Level (SIL), Assurance Level (AL), Software Assurance Level (SWAL), Automotive Safety Integrity Level (ASIL), etc. They require a Certification/Approval process, which expands along a determined life cycle. This is one of the premises for the development of our ES: it must work with criticality levels.

The life cycle model is essential for defining the way the project is developed, how it is translated into specific plans, and how the team ensures fulfilment of objectives. One can find out that all software development life cycles keep a series of common processes in a direct or indirect way: requirements, architecture design, implementation, testing and maintenance. These processes are orders or sequences in a series of steps, which require inputs and generate outputs. In addition, we must define the requirements that the product must meet in each phase as a part of the life cycle model; we also have to define the transition criteria, which guarantee that one phase is complete, so the flow may move to the next one. In the end, the user of our proposed ES will define the

specific life cycle model based on the criteria, which best fit the characteristics of the project: that model must be properly described in the corresponding documentation.

It is necessary to clarify that the scope of each standard or guidance is different, as well as the way how each one approach the requirements. For example, DO-178C [2] and DO-278A [3] do not specify requirements but objectives and activities. The same applies to the concept of validation: in the case of these standards is not used because the system requirements validation is not usually part of the software life cycle processes, although the high-level software requirements must comply with system requirements to help system validation. However, in the case of standard EN-50128 [22], the term validation is also used for software. In other cases, the standard addresses hardware or safety-related processes and they are not specific for software. Once each of the processes applicable in each case has been identified, its inputs, outputs, activities, objectives, requirements, (or what corresponds in each case) can be modelled within an environment, which allows evaluating the compliance of these standards.

The life cycle also includes the necessary activities of verification and validation (V&V), which popularly Boehm [26] expressed through two questions: one for verification (''are we building the product right?'') and validation (''are we building the right product?''). The key element within the inspection models is the use of checklists [27]. This will be the basis for our ES. As previously explained, the term V&V is not used in all standards, but the ES will be designed to be transparent to these two concepts. It will evaluate processes (verification, validation, development, etc.). The concept and scope of each of these processes will depend on each standard and will be defined previously.

### B. AI IN CRITICAL SYSTEMS

AI has been used in Critical Systems in varied system domains; Aviation, Transportation Systems (rail, automotive, etc.) [28], Aerospace Industry [29], Nuclear Systems [30], Medicine [31] among others. AI has provided support for some activities within systems development like design processes [32], testing activities [33] or diagnostic systems [34] in the shape of different techniques: Expert System [35], Fuzzy Logic [36] or Artificial Neural Networks [37]. The goal was always making easier and more effective the different complex activities involved in the development of these systems. AI has been successfully introduced in the Aviation industry some years ago [38] both in the civil and the military fields. Its use has been extended to varied domains ranging from Navigation Systems and Diagnosis and Monitoring Systems to Flight Management Systems, production cost reduction, Pilot Assistance Systems, etc. [39]. We want to add an additional contribution of AI to the development of critical systems in our work: aid to the implementation of specific development practices coming from the standards DO-178C [2] and DO-278A [3] for Airborne and Ground System environments.

### C. DO-178C AND DO-278A – GENERAL OVERVIEW

The standard DO-178C [2], represents a guide for the software development for airborne systems. It is considered by the Federal Aviation Administration (FAA) [40] and European Aviation Safety Agency (EASA) [41] as an acceptable mechanism to demonstrate compliance with applicable Airworthiness regulations for the software aspects of airborne systems and equipment certification. This standard defines five software levels. The software level of a software component is based upon the contribution of software to potential failure conditions as determined by the system safety assessment process [13], [14] by establishing how an error in a software component relates to the system failure condition(s) and the severity of that failure condition(s). The software level establishes the rigor necessary to demonstrate compliance with this standard.

Level A is the most rigorous among the existing five software levels (A, B, C, D and E). Each of these levels establishes a set of objectives to be met (Table 1).

**TABLE 1. Software level and number of objectives DO-178C.**

| Software Level | Effect of Anomalous Behaviour | Number of Objectives |
|---|---|---|
| Level A | Catastrophic failure condition for the aircraft | 71 |
| Level B | Hazardous failure condition for the aircraft | 69 |
| Level C | Major failure condition for the aircraft | 62 |
| Level D | Minor failure condition for the aircraft | 26 |
| Level E | No effect for the aircraft | No objectives |

The standard DO-278A [3] is oriented to not airborne software (ground systems). Currently, the use of this standard is not recognized as an acceptable means of compliance for the airworthiness regulations. Its processes are very similar to DO-178C and it also works with its own set of objectives to be fulfilled (Table 2) related to Assurance Levels. DO-278A has a lengthy section on commercial off-the-shelf (COTS) software that does not exist in DO-178C, but the DO-278A approach can be applied to a DO-178C project.

Both standards [2], [3] considerer the following processes:

1) Software Planning Process.
2) Software Development Process.
3) Software Verification Process.
4) Software Configuration Management Process.
5) Software Quality Assurance Process.
6) Certification/Approval Liaison Process.

The two standards result in the generation of an output data set (Table 3), described in section 11 of both documents [2], [3].

The analysis of these standards suggests a large number of artefacts that must comply with specific objectives and activities distributed in a set of specific processes. They also need to be evidenced before the certifying and approval authorities. This entails generating a knowledge base in which all the artefacts are clearly unidentified.

**TABLE 2.** Software level and number of objectives DO-178C.

| Assurance Level | Effect of Anomalous Behaviour | Number of Objectives |
|---|---|---|
| AL1 | Failure of a CNS/ATM [(1)] system function resulting in a catastrophic failure condition for the aircraft. | 71 + 14 (COTS [(2)]) |
| AL2 | Failure of a CNS/ATM system function resulting in a hazardous failure condition for the aircraft. | 71 + 14 (COTS |
| AL3 | Failure of a CNS/ATM system function resulting in a major failure condition for the aircraft. | 69 + 14 (COTS) |
| AL4 | It is not associated with any failure condition category. | 62 + 14 (COTS) |
| AL5 | Failure of a CNS/ATM system function resulting in a minor failure condition for the aircraft. | 46 + 14 (COTS) |
| AL6 | Failure of a CNS/ATM system function with no effect on aircraft operational capability or pilot workload. | 26 + 14 (COTS) |

[(1)] CNS/ATM: Communication, Navigation Surveillance and Air Traffic Management
[(2)] COTS: Commercial-Off-The-Shell

**TABLE 3.** Items from DO-278A and DO-178C.

| DO-178C/ED-12C | DO-278A/ED-109A |
|---|---|
| PSAC (Plan for Software Aspects of **Certification**) | PSAA (Plan for Software Aspects of **Approval**) |
| SDP (Software Development Plan) | SDP (Software Development Plan) |
| SVP (Software Verification Plan) | SVP (Software Verification Plan) |
| SCMP (Software Configuration Management Plan) | SCMP (Software Configuration Management Plan) |
| SQAP - (Software Quality Assurance Plan) | SQAP - (Software Quality Assurance Plan) |
| Software Requirements Standards | Software Requirements Standards |
| Software Design Standards | Software Design Standards |
| Software Code Standards | Software Code Standards |
| Software Requirements Data | Software Requirements Data |
| Design Description | Design Description |
| Source Code | Source Code |
| Executable Object Code | Executable Object Code |
| SCVP - (Software Verification Cases and Procedures) | SCVP - (Software Verification Cases and Procedures) |
| SVR - (Software Verification Results) | SVR - (Software Verification Results) |
| SECI - (Software Life Cycle Environment Configuration Index) | SECI - (Software Life Cycle Environment Configuration Index) |
| SCI - (Software Configuration Index) | SCI - (Software Configuration Index) |
| Problems Reports | Problems Reports |
| Software Configuration Management Records | Software Configuration Management Records |
| Software Quality Assurance Records | Software Quality Assurance Records |
| SAS - (Software Accomplishment Summary) | SAS - (Software Accomplishment Summary) |
| Trace Data | Trace Data |
| **Parameter Data Item File** | **Adaptation Data Item File** |

### D. EXPERT SYSTEMS

Expert System (ES) is a branch of AI that makes extensive use of specialized knowledge to solve problems at the level of a human expert [42]. The design goal of an ES is capturing the knowledge of a human expert relative to a specific domain of knowledge and code it in a computer in such way that expert knowledge is available to the less experienced users [43]. The main architectural elements of an ES are the Knowledge Base, the Fact Base and an Inference engine. ES that deal with deterministic problems are known as rule-based systems, because they conclude information based on a set of rules using a logical reasoning mechanism. A rule can be defined as a logical statement that serves to relate objects. We can distinguish two main parts in a rule: the premise and the conclusion. The rule is usually written as ''IF **Premise** then **Conclusion**''.

The way of description of our object of study, i.e. compliance to applicable standards, suggests that a Rule-Based Expert System is the best choice to support the decisions. In this case, the knowledge base of the ES is fed by the checklists generated for standard compliance checking, which are mapped to all the objectives and activities of the standards DO-178C [2] and DO-278A [3].

Applying ES to quality processes and development standards is not new. We have studied the existing contributions to analyse if our approach is feasible. In 1985 Moore developed an ES for the control of processes at a general level [44]. Gipe and Jasinski (1986) carried out an analysis of the application of an expert system to solve problems in quality systems, resulting in the viability of this application for this purpose [45]. Paladini (2000) does a review of Artificial Intelligence contributions applied to quality evaluation [46]. Liao, Enke and Wiebe proposed in 2004 an expert advisory system for the ISO 9001 quality system [47]. Eldrandaly (2008) proposed an ES to evaluate Software Quality Assurance based in the ISO/IEC 90003 [48]. Liukkonen, Havia, Leinonen and Hiltunen (2011) proposed an ES to diagnose and optimize the manufacturing process [49]. Main common finding from all these studies is that they show the feasibility of evaluating processes through an ES for quality purposes, so our approach is aligned to the existing background in the area.

### E. ONTOLOGIES

An ontology is a formal and explicit specification of a shared conceptualization [50]. According to [51], the idea is to transform information into knowledge using formalized knowledge structures (ontologies) that reference the data using metadata, under a common standardized scheme on some knowledge domain.

The benefits of using ontologies proposed can be summarized as follows [52]:

- Prevents ambiguity of terms.

- Helps sharing common understanding of the structure of information among the users.
- Enables reuse of knowledge.
- Analyses the domain knowledge.
- Enables the merging of already existing, knowledge, thereby expanding it further.

The objective of this research is not to develop an ontology although we present the basis for extracting knowledge from the standards to be formalized as a data structure. The process is not automated yet, but automation will be proposed as future research work to complement the present research.

In this research [53] an example of conceptualization is presented, where an environment called Domain Specific Engineering Environment (DSEP) is developed: it enables systematic capture of functional domain knowledge. The reusable knowledge in the application and the robotics domain is captured using ontologies. The knowledge capture and reuse mechanisms reduce the time and dependence on external researching to gather a good variety of robotics domain knowledge.

This research [54] presents a requirements elicitation tool called ElicitO aimed at empowering requirements analysts with a knowledge repository that helps in the process of capturing precise non-functional requirements specifications during elicitation interviews.

This tool helps to automate the process of identifying Non-Functional Requirements (NFRs) relevant to a certain domain. This tool helps to automatically identify requirements, while all the requirements are already specified in our research. They only need to be manually captured.

In [55] the author proposes a generic methodology to conceptualize and formalize the tacit knowledge of experts within a team in a collaborative way. It adopts a multilevel approach to generate the core ontology describing generic concepts, and the domain specific ontology representing the experts' skills.

The proposed methodology is based on the work by Fernandez [56], which combines a set of stages and strategies to build ontologies with the following steps:

- **Step** 1 **Specification:** the goal of the specification phase is to produce either an informal, semi-formal or formal ontology specification document written in natural language**.**
- **Step** 2 **Knowledge Acquisition:** using brainstorming, interviews, formal and informal analysis of texts, and knowledge acquisition tools.
- **Step** 3 **Conceptualization:** the domain knowledge will be structured in a conceptual model that describes the problem and its solution in terms of the domain vocabulary identified in the ontology specification activity.
- **Step** 4 **Integration:** The goal is to integrate as many as possible existing ontologies in your ontology.
- **Step** 5 **Implementation:** It is focused on implementing the conceptual model into a formal language like Ontolingua, Resource Description Framework Schema (RDF/S), or Web Ontology Language (OWL).
- **Step** 6 **Evaluation:** It is basically a technical judgment of the ontologies.

In our research we propose the bases to develop a conceptual model of the regulation under study.

## III. METHODOLOGY

We have followed the methodology proposed by Peffers *et al.* [57]. This methodology is based on the Design Science Research (DSR) paradigm, having its roots in the science and engineering of AI. Design research uses design as a method for investigation [58], with the objective of creating ''solutions to specific classes of relevant problems by using a rigorous construction and evaluation process'' [59]. It is fundamentally a problem-solving paradigm [60]. Furthermore, DSR paradigm is highly relevant to Information Systems (IS) research because it directly addresses two of the key issues of the discipline: the central role of the IS artefact and the importance of professional relevance of IS research.

Thus, DSR provides clear, consistent definitions and guidelines for the research process. The defining feature of DSR lies within the construction of new and innovative artefacts that solve relevant design problems. ([61], [57]).

Useful knowledge (in the form of an artefact) could be divided into two categories: descriptive knowledge and prescriptive knowledge [62]. The iterative process gives the opportunity of refining the artefacts and models but also for developing theory [63]. The theories behind DSR belong to either of two categories: descriptive and prescriptive. Descriptive ones, also called kernel theories, frequently have its origin within other disciplines, while the prescriptive ones, the design theories, are prescriptions of ''how to do something''.

Thus, one of the strengths of DSR is the iterative process rendering the theory, which allows to understand better the problem [61], since the same method is used for development and evaluation. The refinement of the designed artefact is aligned to the development of the theory [64]. A method artefact, according to Gregor and Hevner [65] is prescriptive, providing ''the instructions for performing goal-driven activities''.

The model DSR is an adaptation of a computable design process model developed by Takeda, *et al.* [66]. Even though the different phases in a design process and a DSR process are similar, the activities carried out within these phases are considerably different [67]. Both design science research and design [66] use abduction, deduction, and circumscription but there is difference in how these cognitive processes are used.

For their part, Hevner *et al.* [61] proposed a set of principles for doing design science research and established a set of quality attributes for design products, emphasizing the technical quality of the device and the device to achieve the intended organizational purposes. Following this line of thinking, Baskerville, *et al.* developed a set of evaluation criteria for DSR [68]. The evaluation should be formative in the early stages of the project, what allows changes in

the problem and in the nature of the artefact and, even if necessary, redirecting the design [69].

Hevner *et al.* [61] understand the notion of artefact in terms of a construct, a model, a method, and / or an instantiation. On the contrary, March and Smith [70] considered the Design Research strives to create things that serve human purposes. Besides it suggests that a feasible artefact is relevant to a context which can successfully address a problem. Hevner and Chatterjee [71] discuss several benefits of using the Design Research approach. They state that the goal of Design Research is utility as such and that the continuous building and evaluation of artefacts accomplish this. Kuechler and Vaishnavi [72] have extended the general design cycle by reflecting on DSR purposes. All frameworks follow the high-level pattern that is shown in Figure 1 with various degrees of explicit cycles and various levels of defined research outputs.

Peffers *et al.* [57] proposed a methodology that provides a framework for performing DSR. First, the researcher must identify the problem and the motivation, defining the objectives to achieve. After which, the researcher should design and develop an artefact to reach and check the proposed objectives. In the next phase, the researcher should demonstrate the resulting process model through experimentation, simulation of a case study. Then the researcher evaluates the process model using user's satisfaction surveys and user's feedback. Finally, the researcher should show the results.

As described previously, the framework of Vaishnavi and Kuechler [73] has been the choice for our research project. Figure 1 illustrates the different steps in the framework with the different sections and phases of the research.

This approach describes the design process in five steps:

- Firstly, the researcher should identify the problem and the motivation to understand the context of the problem/problem area and so define the objectives to achieve.
- As a second step, the researcher should consult a set of sources, both industrial and academic, for the suggestion of a design, which allows to check and meet our objectives with a contribution.
- The third step is the development of an artefact implemented for the suggested identified solution which is best fitted to the business domain.
- The fourth step is the evaluation, which may be based on as a formal evaluation, on test activities, on simulation, on usability studies and/or on a case study: it will provide clear performance measures.
- Finally, there is a conclusion linked to the assessment of the overall results of the design process, which communicates the results.

## IV. PRACTICAL DESIGN AND DEVELOPMENT

Following the proposed methodology and having identified the problem in the introduction, the development of an ES that facilitates the use and learning of these standards has been determined as our objective and motivation.

The design of the ES is based on a flexible architecture allowing easy reconfiguration of parameters for the use of

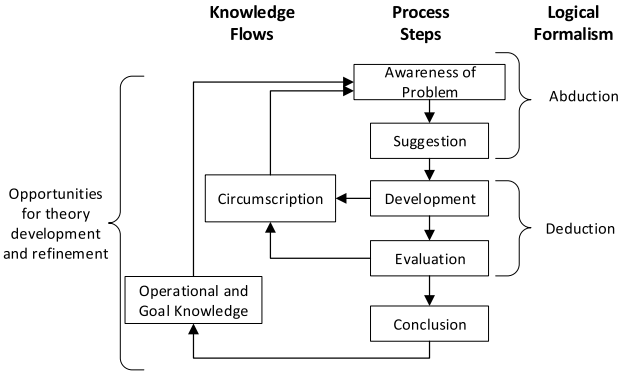



**FIGURE 1.** High level design science research process (adapted from [67]).

both above mentioned standards [2], [3] or even others, which are based on the same concepts. This ES uses the checklists defined in previous works [74], which are part of the knowledge base.

### A. SYSTEM ANALYSIS

Following the proposed methodology, the second step will provide a method for evaluating any life cycle as the use and the object of the application of these standards are oriented towards software development; this will not imply changes in the development of the ES.

The standards [2], [3] describe the different processes shared by most of the life cycles and the interactions among them. Our ES software should be designed to evaluate processes independently of how they are integrated into the selected life cycle. Each process, clearly identified, will have its inputs, outputs and activities. Each of these processes has been systematically represented, as shown in Figure 2.

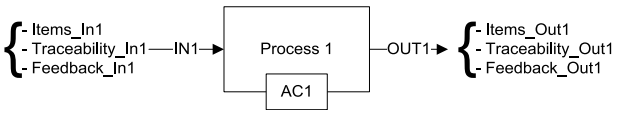



**FIGURE 2.** Unit of process.

Each of these process units can lead to different combinations, which may shape different life cycles. Figure 3 shows two possible examples.

Each process identifies:

- Inputs (IN), which may consist of a series of documents or feedback from other processes.
- Activities, which also provide traceability among processes.
- Outputs (OUT), which, as in the case of the inputs, obtain similar elements.
- Acceptance Criteria (AC) used to determine if a process can be considered completed or requires rework and new review.

All this set of inputs, outputs, activities and acceptance criteria are highly dependent on the applicable criticality

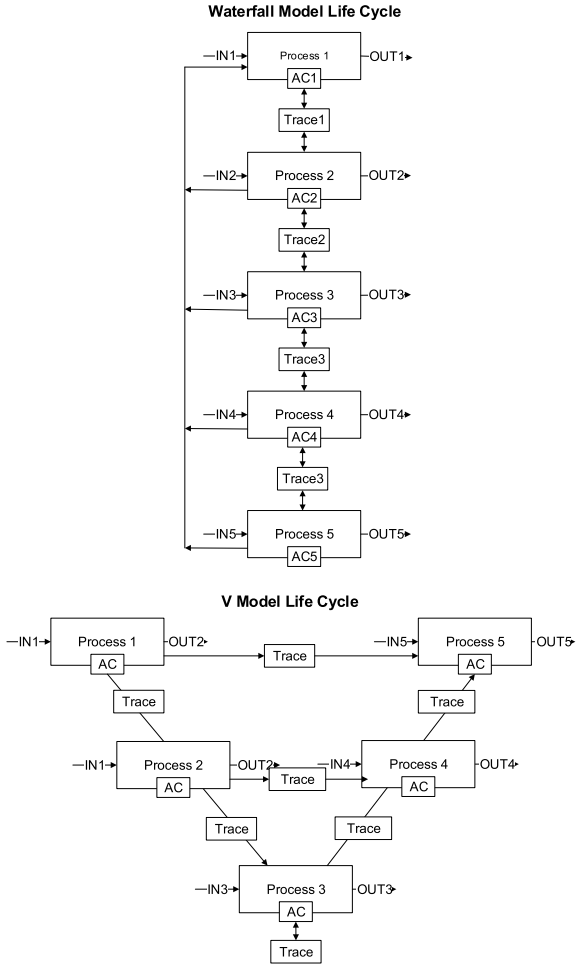



**FIGURE 3.** Software life cycle.

level. The ES will have to deal with this as one of the most important factors.

The standards [2], [3] have been broken-down into a set of processes that are related to each other following this previously defined structure.

Figure 4 shows the evolution of the software life cycle presented by Alessandro Nicoli [75] from DO-178B to DO-178C. It represents a proposal for the adaptation of the processes of the standard DO-178C [2] towards a model that later will be integrated into the ES. The processes of the standard DO-278A [3] are similar to those proposed in DO-178C [2], except that for DO-278A is included an extensive section on the use of COTS software in CNS/ATM systems.

Figure 4 shows the relation among all the identified processes, its dependencies, traceability and items generated. It also includes external processes such as systems engineering, where process defines the configuration items (Software and hardware), systems requirements and Item Development Assurance Level (IDAL). The safety process is also included as an external process. For example, in the case of derived requirements, they must be evaluated by the system safety experts. Other important information shown there are all the traceability between processes. This point is very important as it allows to trace one process to another. The processes are numbered, and this facilitates their implementation in the ES.

The life cycle development processes of these standards are divided into three main types.

- Software Planning Processes.
- Software Development Processes.
  - Software Requirement Process.
  - Software Design Process.
  - Software Code Process.
  - Software Integration Process.
- Integral Processes.
  - Software Verification Process.
  - Software Configuration Management Process.
  - Software Quality Assurance Process.
  - Software Liaison/Approval Process.

Some of these processes (e.g., the verification process) have been divided into other sub-processes to facilitate the evaluation process: the decision was to create a verification sub-process for each development process.

From right to left is the logical sequence of how the processes are executed is shown, except for the integral processes (lower part of figure 4, applicable to all phases of development). From the bottom-top, each of the outputs that are generated in each process are shown. All these artefacts have been traced to process units that finally form the life cycle that implements the ES.

### B. SYSTEM DEVELOPMENT

The second step of the proposed methodology requires the collection of the knowledge, which will be the basis for the ES. The knowledge will have to be recorded and updated in the system, so the simpler method the better. The method for extracting and shaping the knowledge for the system must be detailed.

In the particular case of this research, we propose the basis for transforming the content of a specific standard into a formalized knowledge structure with the aim of representing knowledge in a specific way that can be used by the ES. This scheme of knowledge structure raises the bases of an Ontology. The followings steps have been taken to obtain the formalized knowledge structure:

- **Step 1:** Obtaining the DO-178C and DO-278A process map.
- **Step 2:** Identification of the set of requirements applicable to of each process.
- **Step 3:** Generation of the checklists.
- **Step 4:** Verification of the requirements.
- **Step 5:** Generation of the set of rules that form the knowledge base.

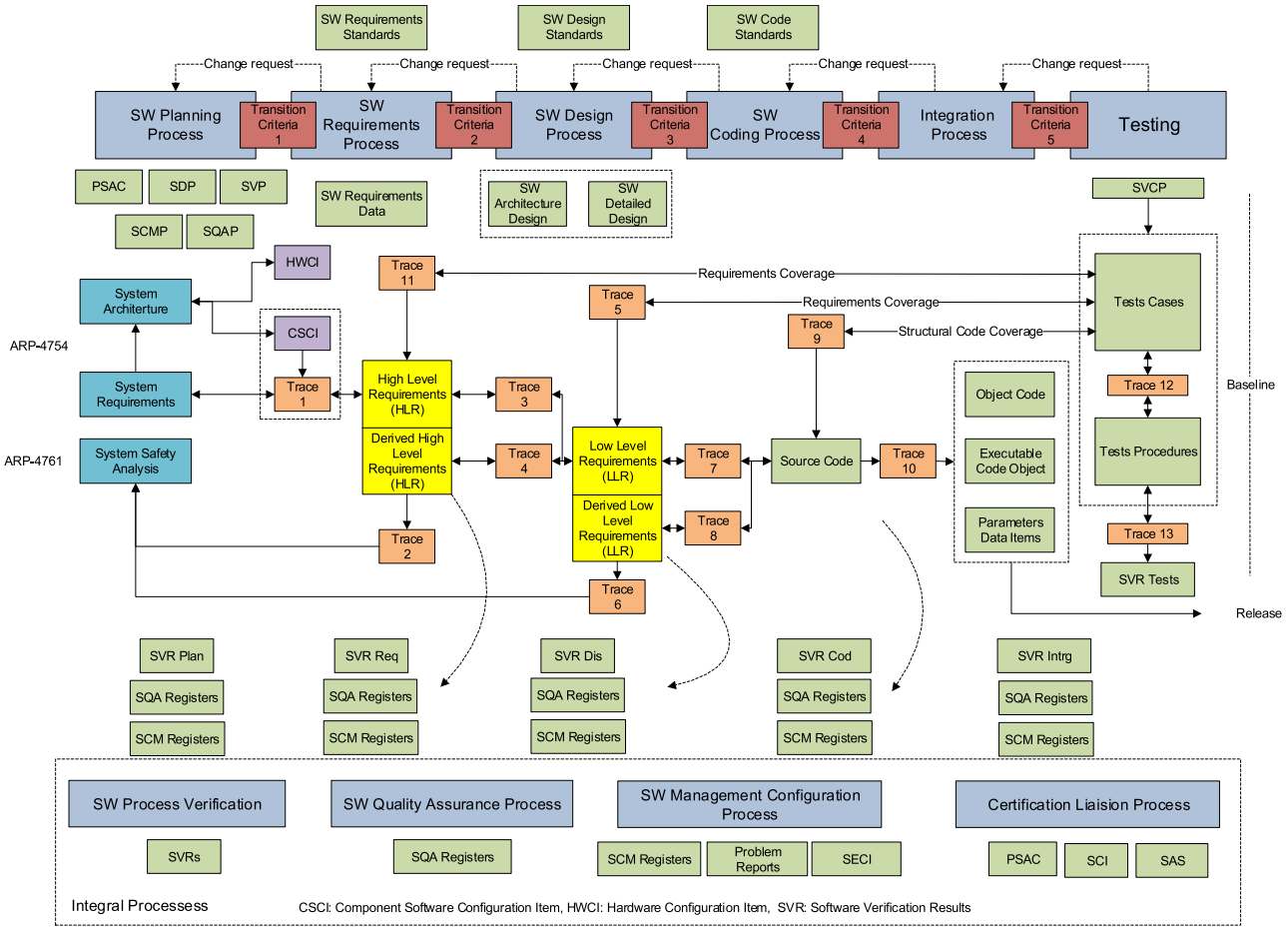



**FIGURE 4.** Software life cycle proposal software life cycle DO-178C.

- **Step 6:** Generation of a standardized data structure (excel sheet) that later becomes an XMLS.

### 1) STEP 1. OBTAINING THE DO-178C AND DO-278A PROCESS MAP

Our ES has been designed to cover all processes identified by the standards [2], [3], thus providing enough support in the evaluation of the compliance of each process. It will facilitate the task of finding out if each of the objectives identified by each level of the application is met (DAL or AL).

The method to evaluate the software development based on these standards is reduced to a simple process of evaluating the inputs of each process, its activities and objectives, and the outputs and transition criteria. The first required information is defining the criticality level for the software (DAL for DO-178C and AL for DO-278A). With these first data, the large majority of objectives and activities to be carried out are already defined. With these data, we can get other additional information such as:

- Each of the processes of the life cycle.
- Each of the objectives and activities to be carried out in each of the processes.
- The input elements required for each process.
- The necessary independence. (separation of responsibilities to ensure objective evaluation).
- The roles required for the development.
- The input and output items required.
- The Transition Criteria.

All this basic information is necessary to define and detail processes in the first phase. It has been defined as Preconditions. They are the low-level details that the user should know at the beginning of a development and that allow to break-down and define the scope of certain activities. For example, Additional Conditions like the need for tool qualification, use of COTS etc.

The identification of all processes is detailed in Figure 4.

### 2) STEP 2. IDENTIFICATION OF THE SET OF REQUIREMENTS APPLICABLE TO OF EACH PROCESS

All processes have been tagged in a specific order from Process_01 to Process_14. The processes are mapped to each specific section of each standard and all objectives and activities are tagged.

Example of DO-178C [2]:

- Process_01; Software Planning - SPPxxx
- Process_02; Software Development - Requirements - SDPxxx

- Process_03; Software Development - Software Architecture - SDPxxx
- Process_04; Software Development - Software Detailed design - SDPxxx
- Process_05; Software Development - Software Coding - SDPxxx
- Process_06; Software Development - Software Integration - SDPxxx
- Process_07; Software Verification - Software Requirements - VOSRPxxx
- Process_08; Software Verification - Software Design - VOSDPxxx
- Process_09; Software Verification - Software Coding - VOSCIxxx
- Process_10; Software Verification - Software Integration - TOIP002xxx
- Process_11; Software Verification - Software Verification - VVPR001xxx
- Process_12; Software Config Management - SCMP001xxx
- Process_13; Software Quality Assurance - SQAP001xxx
- Process_14; Certification Liaison - CLP001xxx

Where xxx indicates the number of requirements mapped. This is very useful in the process of verification.

#### 3) STEP 3. GENERATION OF THE CHECKLIST

We have already identified all the objectives and activities of both standards and created the corresponding set of checklists to evaluate compliance with them, covering all the processes, in our previous work [74]. These checklists along with other aspects of the standards have been transformed into rules for the ES: sometimes they are not strictly objectives or activities, but they have been considered relevant to demonstrate compliance. We have considered the material included in Developing Safety-Critical Software [76] by Rierson as a good reference. All the processes of DO-178C are very detailed and it is a good complement to have a solid knowledge data base. Another reference is Avionics Certification [77] by Hilderman & Baghai, has been analysed and also considered a good reference for the knowledge of this standard. It should be noted that each of the standards [2], [3] has been independently analysed.

#### 4) STEP 4. VERIFICATION OF REQUIREMENTS

This verification is carried out to check that all the objectives and activities of the DO-178C and DO-278A have been correctly identified and that they cover the whole domain. In section B VERIFICATION (Verification of Knowledge base (Representation of the Domain)), this process is explained in more detail.

#### 5) STEP 5. GENERATION OF THE SET OF RULES THAT FORM THE KNOWLEDGE BASE

The process for transforming these checklists into production rules is the following one:

- Each objective and activity of the standard is mapped to a checklist.
- For each checklist, at least a set of three rules are defined:
  - Rule 1: Indicate what to do to the user based on initial preconditions.
  - Rule 2: If the user does not perform the action, it shows ''the how''.
  - Rule 3: If the user performs the action, it shows ''how to evidence it''.

We show below an example of how a checklist is created from an objective of the standard in the shape of a set of rules:

- Objective: 4.1.a DO-178C
- Checklists: Are the activities of the software development processes defined?

This checklist is represented in the fact SPP1 (Software Planning Process) Rules:

- **RULE 1**: **IF** Objective_A_1_1 = YES **THEN** Define the activities of the life cycle development software processes. According 4.1.a DO-178C paragraph.
- **RULE 2**: **IF** Objective_A_1_1 = YES **AND** SPP1 = NO **THEN** The activities should be defined in the PSAC, SDP, SVP, SCMP and SQAP plans.
- **RULE 3**: **IF** Objective_A_1_1 = YES **AND** SPP1 = YES **THEN** Verify that the activities are defined in the plans PSAC, SDP, SVP, SCMP and SQAP. Include the results of this verification in the form of a Software Verification Record (SVR).

The ES allows to show, for each conclusion of each rule, a help message in text form that allows to understand the objective or activity through an explanation module. For example, for RULE 1, the following text is showing:

- Objective 4.1.a – DO-178C: The standard distributes the activities for the following processes:
  - Software Planning
  - Software Development
  - Software Verification
  - Configuration Management Software
  - Software Quality Assurance
  - Software Liaison/Approval Process

Each standard [2], [3] show the applicable objectives for each level of criticality (DAL and AL) in tables in its Annex A. These tables provide a reference for each objective, which is described in more detail in the document. These objectives are usually expressed as SHOULD statements because the content of these standards is the recommendation, and the terms ''shall'' or ''must'' are not included. For each one of these objectives, we have defined a checklist and a set of at least 3 rules, as shown above. All the generated checklists with rules are integrated into the ES within the database: they are activated through a set of rules, which will be in turn activated according to a set of preconditions.

#### 6) STEP 6. GENERATION OF A STANDARDIZED DATA STRUCTURE (EXCEL SHEET) THAT LATER BECOMES AN XMLS

Following this process, the content of the standards [2] and [3] are transformed into a standardized data structure that forms the knowledge base. The set of rules and facts obtained are contained in a spreadsheet (XLS) file. Through a process that is transparent to the user, it is converted into an XMLS format when the file is loaded by the ES. Users with little knowledge in Artificial Intelligence will be able to add new rules or update the existing ones using this format instead of doing it directly in the XMLS file.

JRuleEngine has been selected as inference engine based on Java Specification Request 94, release 1.1 [78], considered as a sophisticated IF/ELSE statement interpreter, which acts on input objects (facts) to produce output objects (conclusions or inferences). It is based on a direct chaining algorithm using an XMLS syntax. It is freeware and uses an open source language something very important for our study.

Completing the ES requires the definition of a set of additional rules to define the behaviour of the ''pre-conditions'': these rules are activated depending on a particularization of the process. If they come true, the corresponding process rules become applicable; otherwise they remain inactive. We show below two examples of precondition;

- **RULE 4:IF** DAL = A **THEN** Objective_A_2_4 = YES. Where ''DAL = A'' is the precondition, when DAL selected is A the objective A-2-4 is applicable for the whole process.
- **RULE 5:IF** Objetive_A_1_4 = YES **AND** Multiversion = YES **THEN** Activity 4.2.f. Table A-1. Define the methods and tools necessary to achieve the necessary dissimilarity to meet the safety objectives of the system. Where ''Multiversion = YES'' is the precondition. When this precondition is selected it is necessary to include additional considerations to the process (Activity 4.2.f. of DO-178C).

  The word ''Multiversion'' represents a fact in form of checklist that the ES requests to know at the beginning of the evaluation of a process and that it is introduced by the user through the Interface Human Machine (IHM).

The results of each checklist allow to activate different rules, and, at the same time, they are translated into new conditions for another checklists and rules. All the generated rules have been distributed along with the different processes that each standard defines and which in turn must be interrelated (Figure 4). The set of rules for each process (with objectives, activities, inputs, outputs and transition criteria) sets up the units of process (Figure 2) and these units are related among them (Figure 3) shaping the life cycle.

The transition criteria are used to determine if a process can be considered complete or must be reviewed again. Each process of the software life cycle is associated with activities, which work on inputs to produce outputs. A process can produce a response to other processes and receive a response from others. These criteria have been defined in two parts: input transition criteria and output transition criteria. The transition criteria will depend on the planned sequence of software development processes and integral processes of the life cycle. They may be affected by the criticality level for the software. Each of these transition criteria will generate a set of specific rules, which will allow their control and monitoring. In Appendix A of Developing Safety-Critical Software [76] is shown a good example of how to treat the Transition Criteria and they have been considered for this ES.

### C. DEVELOPMENT OF THE INTERFACE AND PROPOSED ARCHITECTURE

Following the proposed methodology, the third step is the development of the artefact, which is an ES. The developed ES must have a certain degree of flexibility within the rigidity of specificity. This flexibility should allow applying different standards of varied nature to the same project, always depending on the needs and preferences of the organization that is using the system.

The ES integrates a set of dashboards to show the user the evolution and compliance of each process, Figure 6. It will show for each process if the objective is satisfied or it is still pending (or maybe it is not applicable).

Open source Software technologies are the basis of development of the ES as they reduce the final cost of development [79], [80]. Java is the chosen programming language adding two specific libraries for the implementation. The first is JRuleEngine as inference engine. The other is JFreeChart, which offers different tools to implement all types of diagrams and graphical elements, very helpful for the preparation of monitoring reports and dashboards. MySQL is the underlying database management system selected while MySQL Workbench desktop software helped in the management of tables and the control of the application data.

As the ES can be configured to be used with different regulations and standards, the system includes configuration files allowing a flexible configuration environment. The block diagram in Figure 5 shows this flexibility. The knowledge base of the system is nurtured by several configuration files in xlsx format (Microsoft Excel). These files are imported and converted to an XMLS format within the ES. The system works with two types of configuration files:

- A global configuration file, where all the processes and their inputs and outputs are defined for the project to be evaluated.
- A configuration file for each process, which defines:
  - The rules to be applied by the inference engine.
  - The variables to be handled by those rules according to the phase of the process being evaluated (preconditions, activities, transition criteria, etc.)
  - The texts that the explanatory module must show when the rules are fulfilled.

During the import of these configuration files, the system processes overall information and stores it in its knowledge base, represented by the database. The outputs of the

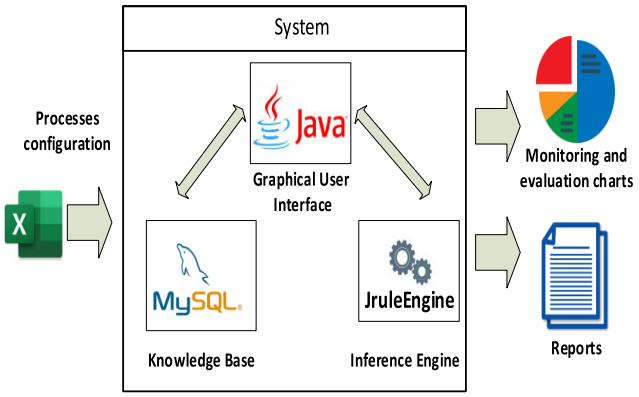



**FIGURE 5.** Block diagram of the system.

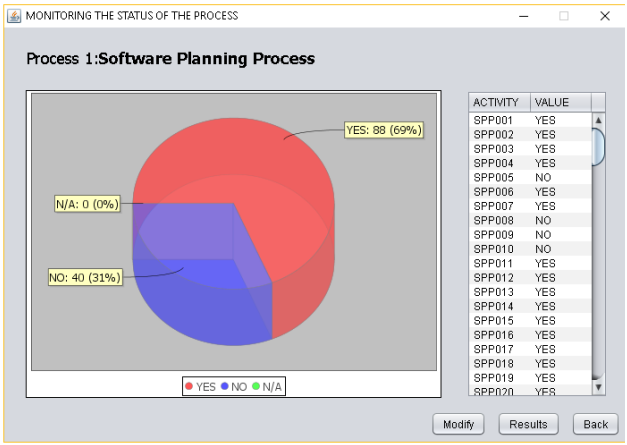



**FIGURE 6.** Example of Dashboard for a process.

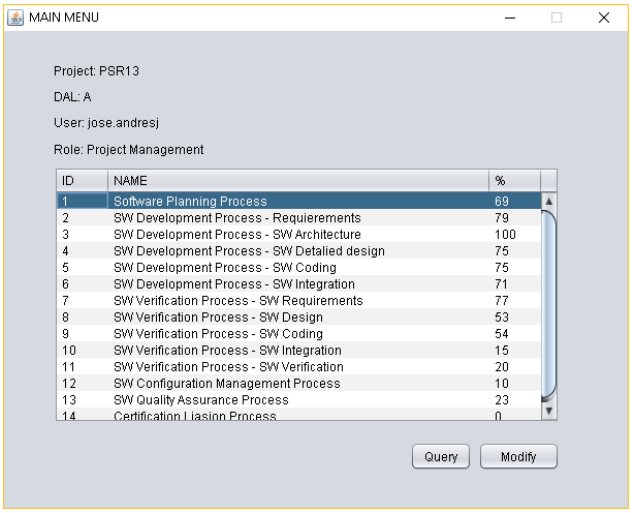



**FIGURE 7.** Example DALA – Process DO-178C.

application correspond to a series of reports generated in real time by the system during the validation of the project processes. The flexibility of these outputs resides in the fact that, independently of the standard that is being applied to perform the evaluation of compliance for the project, the user may have the option of a dashboard to monitor the status of each of the processes. So, it is possible to generate reports

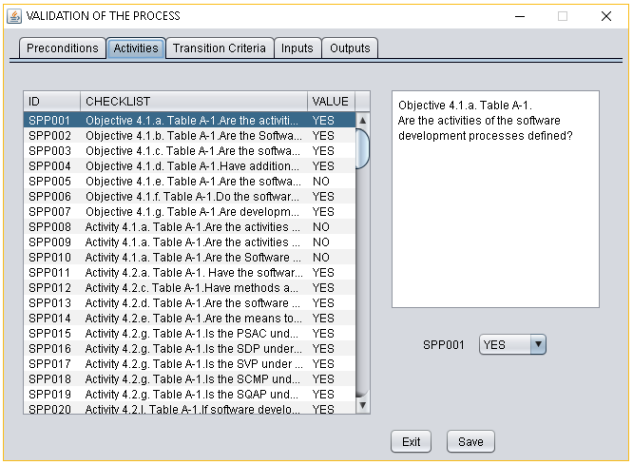



**FIGURE 8.** Example DALA - Objectives / Activities - DO-178C.

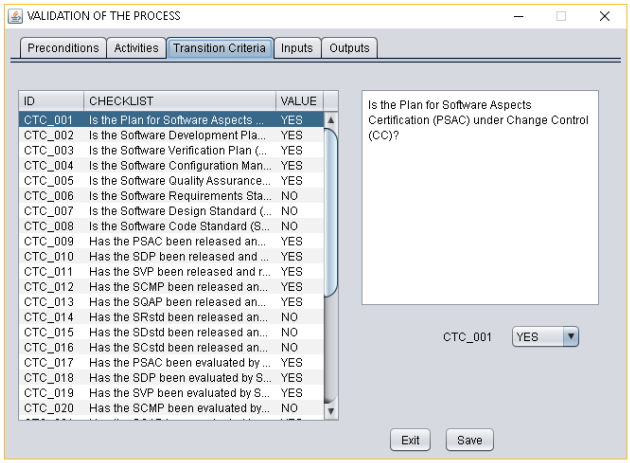



**FIGURE 9.** Example DALA - Transition Criteria - DO-178C.

collecting all the recommendations and corrections necessary to optimize the project under analysis: this depends on the rules, which have been fulfilled during the validation of the processes. This configuration scheme also adds efficiency to the system as it provides an easy way of adding, modifying and eliminating knowledge items.

The following figures show some examples of the results offered by the ES.

## V. ANALYSIS AND DISCUSSION OF THE RESULTS

Following the proposed methodology, the fourth step is the evaluation. In this section, we describe the process of validation and the verification of compliance.

### A. SYSTEM VERIFICATION AND VALIDATION

During the development of the ES we have considered the main problems, which may impact it and we adopted a strategy to mitigate them: errors such as the incomplete transfer of knowledge to the ES, errors in the rules that form the knowledge base; redundancy, inconsistency, incompleteness and the occurrence of cycles [81]. However, the ES must be verified and validated to gain confidence in fulfilling the

required functionality and characteristics. The evaluation of our ES requires V&V but contextualized to an ES:

- Verification will check if the ES is logically consistent, but it does not guarantee that its domain-dependent knowledge agrees with the one coming from the human expert [82].
- Validation is the process of evaluating the ES during and after the development process to ensure that compliance results are the same as the ones got by experts [82].

### B. VERIFICATION

#### 1) VERIFICATION OF KNOWLEDGE BASE (REPRESENTATION OF THE DOMAIN)

It has been verified that the domain of knowledge is sufficiently identified in the knowledge base. Part of this domain is perfectly bounded since the designed ES focuses on very specific standards [2], [3]. These standards, in turn, identify very clearly the aspects of compliance to be checked. This verification was carried out by tracing the standards to the elements of the knowledge base and verifying traceability. All the objectives and activities of the standards are traced to rules that belong to the knowledge base, so the knowledge that represents the evaluated domain is correctly transmitted from the standards to the rules of the knowledge base.

#### 2) VERIFICATION OF KNOWLEDGE BASE (CONSISTENCY AND COMPLETENESS)

The number of generated rules is very high (more than 3000 for each standard), so it is necessary to verify that the developed rules are error free [83]. We applied different verification techniques [84] to reduce the probability of introducing structural errors in the knowledge base. We considered the following checks to ensure the consistency and integrity of the rules:

- **Redundant rules**: rules with identical premises and conclusions.
- **Conflicting rules**: rules with identical premises, but with conflicting results.
- **Circular rules:** set of rules, which can create an infinite loop in its execution.
- **Unnecessary IF conditions:** two rules with the same conclusion and one premise in each rule contradicts another while the rest of the premises are identical [85].
- **Contradictory rules:** rules that could not be fulfilled because their premises were contradictory and therefore, always evaluated as false.
- **Missing rules:** facts not used within the inference process, dead-end goals, or situations when all legal values of some input are not present in the premises of the rule set [81].
- **Dead-end rules:** rules that cannot be interpreted by the rule-based system [85].
- **Unreachable rules:** in a general backward-chaining system, a rule is unreachable if it has a conclusion that is not a goal of the system and not a condition of another rule [81].

#### 3) VERIFICATION OF THE USER INTERFACE

The verification will ensure that the user experience with the system is satisfactory. The following features have been verified:

- Navigation on the different windows belonging to the graphical interface of the ES.
- Query and modification of the data of each process.
- Registration and access of users with different roles and permissions.
- Registration of the new projects and processes to include in them.
- Update of the process information.
- Import of the configuration of the processes.
- Load of rules.
- Management of pre-conditions, activities, transition criteria, inputs and outputs.
- Query of the results obtained.

#### 4) VERIFICATION OF THE DESIGN REQUIREMENTS

Verification was focused on some requirements like the following ones:

- Use of free software technologies.
- Architecture of the system is restricted to a local/enterprise network scope.
- Authentication module and hierarchy of roles.
- Well differentiated user profiles: the administrator and the standard user.

#### 5) VERIFICATION OF CONCLUSIONS REACHED

This is the verification that all the possible conclusions that can be reached by the ES are achievable [83]. This will guarantee that the set of proper outputs is reached for a logical range of inputs. This has been verified by checking that, with all the possible combinations of proposed preconditions, the appropriate output is generated.

It is necessary to emphasize that the activities of verification of the knowledge base were carried out by different human experts who built it guaranteeing a degree of independence and objectivity. The human experts in charge of performing the verification have profiles ranging from software quality assurance with more than 10 years of experience in quality control in different industrial sectors and technologies. And Software developers with experience between 2-7 years in the development of Expert Systems.

### C. VALIDATION

The developed ES was tested to ensure that the built model correctly represents the problem of the domain. We selected a set of test cases representing real cases considering several factors such as the applicable software level, the set of possible preconditions and the selection of specific process and roles.

Once established the necessary entry points (preconditions), this set of test cases were tested using two type of tests:

1- Directly entering them through the standard input as a human expert would do without the use of the ES.

The user has been provided with a predefined set of checklists to cover all the processes that conform to the life cycle. Users chose the checklists they considered adequate under their own criteria.

2- Direct use of the ES: in this case, the set of checklists is part of the ES and they are automatically selected by the system.

For both cases;

- The standard is provided as a reference.
- Basic project documentation is provided, the same for the two tests.
- An independent third person answers whether or not the information requested by the evaluators is available. The same answers are provided for the two tests.

These two methods have been carried out by two different types of users: the first one by an expert domain in the use of these standards and the second one by a technical person with no expertise in the use of the standards. After the tests, we compare all the results and collect information through a short satisfaction, usability and accessibility questionnaire to capture the perceived opinion of testers on the check of regulations using the ES.

The expert domain who has validated the SE has more than 10 years of experience in the development and validation of systems based on the DO-178B [86], in the Aerospace software projects. The person with no expertise in the use of this standard (DO-178C) has more of 5 years of experience in the development of embedded systems in different application sectors (railway, aerospace).

The validation of the ES has been carried out on based on a project where its documentation is public, so this allows for the repeatability of the results. A second validation was carried out, taking three private projects as reference documentation.

The documentation repository of the first validation is based on public documentation of a project based on the development of the DO-178B [86], which allows to test the Expert system. While it is a project that does not contain all the information in detail, it is enough to test the Expert system. The base project has the following documentation:

- Guidance and Control Software Project Data-Volume 1: Planning Documents [87].
- Guidance and Control Software Project Data-Volume 2: Development Documents [88].
- Guidance and Control Software Project Data-Volume 3: Verification Document [89].
- Guidance and Control Software Project Data-Volume 4: Configuration Management and Quality Assurance Documents [90].

The Guidance and Control Software (GCS) project was the last in a series of software reliability studies conducted at Langley Research Center between 1977 and 1994 [87]–[90]. Some of the software components used as part of the GCS project were developed to be compliant with the RTCA/DO-178B software standard [86].

This project is a good candidate to be evaluated with our ES. This exercise also gives the possibility to check that specific points of a system based on an older version of the standard can be improved to migrate them to the latest version of the standard. This Project only allows to validate the part of the ES based on the DO-178C [2].

As in the verification activities, the validation of the system was carried out by people different (above mentioned) to those who built it, guaranteeing a degree of independence and objectivity.

A confusion matrix has been used to describe the performance of the ES based on the test data for which the positive (i.e. true) values are known [91]. The basic attributes of a confusion matrix are true positives, true negatives, false positives, and false negatives, respectively. Classification accuracy, sensitivity, specificity, positive predictive value and negative predictive value can be defined by using the elements of the confusion matrix as:

- Accuracy = (TP+TN)/(TN+TP+FN+FP)
- Positive predictive value = TP/(TP+FP)
- Negative predictive value = TN/(FN+TN)
- Sensitivity = TP/(TP+FN)
- Specificity = TN/(FP+TN)

**True positive (TP):** An Activity/objective, precondition or transition criteria marked as "**Yes**" is displayed as "**Yes**" in the validation Screen process and Dashboard, and all items applicable and the output report process are "**right**" in the Expert System.

**True negative (TN):** An Activity/objective, precondition or transition criteria marked as "**No**" is displayed as "**No**" in the validation Screen process and Dashboard, and all items applicable and the output report process are "**right**" in the Expert System.

**False positive (FP):** An Activity/objective, precondition or transition criteria marked as "**No**" is displayed as "**Yes**" in the validation Screen process and Dashboard, and all items applicable and the output report process are "**not right**" in the Expert System.

**False negative (FN**): An Activity/objective, precondition or transition criteria marked as "**Yes**" is displayed as "**No**" in the validation Screen process and Dashboard, and all items applicable and the output report process are "not right" in the Expert System.

Table 4 and 5 show the evolution of the confusion matrix from the first set of tests until the last set of tests performed. Between the first and last test, the ES has been improved after introducing the modification to fix errors in the rules.

- Classif. accuracy (%) = 79,37%
- Sensitivity (%) = 82,45%
- Specificity (%) = 59,00%
- Positive predictive value (%) = 93,00%
- Negative predictive value (%) = 33,71%
- Classif. accuracy (%) = 100%
- Sensitivity (%) = 100%
- Specificity (%) = 100%
- Positive predictive value (%) = 100%

**TABLE 4.** Confusion matrix with the value for every measure – First set of tests.

| Actual | Predicted – First Test | | |
|---|---|---|---|
| | **Positive** | **Negative** | **Total** |
| Positive | True Positive (TP) = 545 | False Negative (FN) = 116 | **661** |
| Negative | False Positive (FP) = 41 | True Negative (TN) = 59 | **100** |
| Total | **586** | **175** | **761** |

**TABLE 5.** Confusion matrix with the value for every measure – After 6 set of tests.

| Actual | Predicted – Last set of testing | | |
|---|---|---|---|
| | **Positive** | **Negative** | **Total** |
| Positive | True Positive (TP) = 695 | False Negative (FN) = 0 | **695** |
| Negative | False Positive (FP) = 0 | True Negative (TN) = 66 | **66** |
| Total | **695** | **66** | **761** |

- Negative predictive value (%) = 100%

The measured parameters correspond to how the system evaluates the different input conditions, such as preconditions, objectives/activities, transition criteria and outputs, being the total of elements evaluated.

Once all the improvements have been made, our system achieves 100% accuracy on a set of 761 input and output parameters. Sensitivity has a high value, which denotes the system's ability to correctly perform the ratio of Objectives/activities, preconditions, transition criteria and items. In the first test, the specificity had a lower value. This can be explained by the fact that our system contained some errors in the rule set, and once corrected this value is increased. An important measure for our system is the true predictive value. With a value of 100% is a very promising result.

The results show that in the first validations, the system included some errors that after several improvements and tests were corrected obtaining the expected results.

Table 6 shows the obtained results when used for evaluating the standard DO-178C. As indicated above, the ES has been tested by two types of users: the first (expert) has evaluated a set of tests without using the ES and the second tester (with no expertise in standards) has evaluated the same case using the ES.

The users evaluated all the established processes assuming the required role in each case; development, verification, validation or quality. In the process of manual validation, the user refers to the standard to select what objectives and activities were necessary to comply with the proposed DAL. Table Annex A [2]. This information in detail is obtained from chapters 4 to 9. The same happened with the items where the user must refer to chapter 11 to establish the correct content that these items should have. When using the ES, all this process is automatic because all these data are automatically selected by the system once DAL is selected.

**TABLE 6.** Validation results with ES-DO-178C (Particular case to validate ES with the selection of DAL A).

| Process | NO ES | | | | ES | | | |
|---|---|---|---|---|---|---|---|---|
| | **Objectives** | **Activities** | **Items** | **Checklist** | **Objectives** | **Activities** | **Items** | **Checklist** |
| Software Planning Process | 7 | 26 | 9 | 124 | 7 | 27 | 9 | 128 |
| Software Requirements Process | 2 | 11 | 2 | 24 | 2 | 11 | 1 | 24 |
| Software Design Process - Architecture Design | 1 | 2 | 1 | 4 | 1 | 2 | 1 | 4 |
| Software Design Process - Detailed Design | 2 | 11 | 2 | 26 | 2 | 12 | 1 | 28 |
| Software Coding Process | 1 | 5 | 2 | 8 | 1 | 5 | 2 | 8 |
| Integration Process | 1 | 6 | 2 | 14 | 1 | 6 | 2 | 14 |
| Software Verification Process - Software Requirements | 7 | - | 1 | 13 | 7 | - | 1 | 13 |
| Software Verification Process - Software Design | 13 | - | 1 | 17 | 13 | - | 1 | 17 |
| Software Verification Process – Coding | 9 | - | 2 | 13 | 9 | - | 2 | 13 |
| Software Verification Process - Integration | 5 | 18 | 3 | 71 | 5 | 18 | 3 | 71 |
| Software Verification Process - Verification of Verification | 9 | 20 | 1 | 46 | 9 | 20 | 1 | 46 |
| Software Configuration Management Process | 9 | 9 | 4 | 67 | 9 | 9 | 4 | 67 |
| Software Quality Assurance Process | 5 | 21 | 1 | 31 | 9 | 21 | 1 | 31 |
| Certification Liaison Process | 3 | 8 | 3 | 19 | 8 | 8 | 3 | 19 |

The same thing happens with the selection of the checklists. During manual validation, the user must select the most appropriate set of checklists for each process, while that is automatic with the ES. In this case, the user experience is more important for manual validation than with the use of the ES. In both cases, the same preconditions were chosen. During the manual validation, the user must establish how they influenced the objectives, activities and items. Again, it is automatic with the ES.

Once these points were determined, the evaluation of each objective and activity was done manually in both cases. While

in the manual validation process the results were registered in an Excel file, with the SE a drop-down menu allows selecting if the associated checklists have satisfactory results (Yes), unsatisfactory (No) or they were not applicable (N/A) by default. Once each test case was evaluated, it was necessary during the manual validation to make the final verification of how many objectives and activities were fulfilled while it is automatic with the ES.

We can observe that the manual validation has overlooked some objectives and activities, and therefore the corresponding checklists have not been used. In the case of the test with the ES, we also detected a case where a specific output was not evaluated. It was later confirmed that the reason was an error in coding rules (once solved, it showed the correct output).

Another finding was that the activities were more disaggregated into sub-activities when using the ES, so it was easier to evaluate them but showing a greater set of activities. It was decided to show the set of activities without breaking them down in order to keep comparable the results of both methods. Apart from this, it is observed that the outputs shown by the test with the ES are the expected ones when compared with the manual test. The selection of objectives also shows correct results.

Table 7 shows an example of the total of the processes, checklists, preconditions, transition criteria and rules used for the case of the standard DO-178C.

Regarding the perception of satisfaction, usability and accessibility of the SE, the opinion was collected from other additional ten users (experts and no experts in the field) with a questionnaire with a 5-point Likert scale for responses [92]. The following questions have been performed:

Q1 Degree of satisfaction with the application

Q2 Degree of efficiency provided by the application

Q3 Overall assessment of the application

Q4 Degree of the application facilitates the use, control and management of the processes.

Q5 In the use of SE applications, you consider that these applications make work easier and motivate.

Q6 Usability: Interoperability with the interface.

Q7 Usability: Transitions between different screens and ordered control system has been valued.

Q8 Degree of accessibility of information.

Q9 Data entry and validation of the system

And Table 8 represents the questions and results obtained.

The degree of satisfaction with the application reached 4.4, the degree of efficiency provided by the application scored 4.7 and the overall assessment of the application was 4.7. The question on the usefulness of the proposed ES to facilitate the use, control and management of processes reached 4.8. The general perception of the ES as a system which makes easier the work and motivates the users reached 4.6. We also included two questions referred to usability: average score for the operabilityy of interface was 4.3, while transitions among the different screens reached a value of 4.8. Finally, questions on accessibility of information scored 4.7 and the system data entry and validation averaged 4.4.

The people who have answered the questionnaire, have varied profiles ranging from computer developers, engineering students in computer science, with experience ranging from 1-10 years, in the development of computer applications. In general, the opinion of the ten users of the ES was positive in all the quality aspects covered by our questionnaire.

**TABLE 7. ES Parameters for DO-178C (Particular case to validate ES with the selection of DAL A).**

| Process | Nº Checklists | Nº Preconditions | Nº Transition Criteria | Nº Rules |
|---|---|---|---|---|
| Software Planning Process | 128 | 26 | 33 | 832 |
| Software Requirements Process | 24 | 4 | 12 | 186 |
| Software Design Process - Architecture Design | 4 | 1 | 10 | 89 |
| Software Design Process - Detailed Design | 28 | 6 | 12 | 214 |
| Software Coding Process | 8 | 3 | 13 | 134 |
| Integration Process | 14 | 5 | 10 | 130 |
| Software Verification Process - Software Requirements | 13 | 2 | 12 | 176 |
| Software Verification Process - Software Design | 17 | 3 | 8 | 211 |
| Software Verification Process - Coding | 13 | 2 | 13 | 209 |
| Software Verification Process - Integration | 71 | 6 | 11 | 382 |
| Software Verification Process - Verification of Verification | 46 | 8 | 18 | 354 |
| Software Configuration Management Process | 67 | 8 | 12 | 363 |
| Software Quality Assurance Process | 31 | 3 | 2 | 155 |
| Certification Liaison Process | 19 | 1 | N/A | 115 |
| Total | **487** | **78** | **166** | **3550** |

**TABLE 8. ES perception of satisfaction, usability and accessibility.**

| Answers 1-5 (1 worst, 5 best) | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 | Q8 | Q9 |
|---|---|---|---|---|---|---|---|---|---|
| User 1 | 4 | 4 | 4 | 5 | 5 | 4 | 5 | 5 | 5 |
| User 2 | 4 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 4 |
| User 3 | 4 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 4 |
| User 4 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 5 |
| User 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 4 |
| User 6 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 4 |
| User 7 | 5 | 4 | 5 | 5 | 4 | 5 | 5 | 4 | 5 |
| User 8 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 4 |
| User 9 | 4 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| User 10 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Mean | 4,4 | 4,7 | 4,7 | 4,8 | 4,6 | 4,3 | 3,8 | 4,7 | 4,4 |

### D. EXPERIMENTAL/USECASE VALIDATION

The selected use cases are based on the evaluation of the two Data Items that are part of the Software Planning process for the DO-178C standard:

SW Requirements Standards.

SW Design Standards.

The objectives/activities related to the process corresponding to the SW Planning Process have been evaluated through the ES with specific focus on these two Data items.

The available project documentation is the input. The Guidance and Control Software Project Data-Volume 1: Planning Document [87] describes the content and scope of development standards in this case. Where necessary, the evaluator has made use of another available project information documents [88]– [90].

A DAL A has been selected as a system precondition. Each of the objectives/activities applied to these two data items and the transition criteria defined for these data items have been evaluated. Each of these two use cases was then checked manually without using the SE leading to conclusions when comparing both results. The examples of use case to validate the Expert System are shown in the Appendix.

Table 9 shows the activities, objectives and checklists used in to validate these two use cases. As can be observed, the results in both cases are identical, both in the use of the SE and in the non-use of the SE. The only difference shown is that in the case of the use of the SE the information that the user had to check has been shown automatically, while in the case of the non-use of the SE, this process has been slower.

**TABLE 9.** ES experimental use case DO-178C - DAL A.

| Process | NO ES | | | | ES | | | |
|---|---|---|---|---|---|---|---|---|
| | Objectives | Activities | Items | Checklist | Objectives | Activities | Items | Checklist |
| Software Planning Process (total) | 2 | 6 | 2 | 20 | 2 | 6 | 2 | 20 |
| Software Requirements Standards | 1 | 3 | 1 | 9 | 1 | 3 | 1 | 9 |
| Software Design Standards | 1 | 3 | 1 | 11 | 1 | 3 | 1 | 11 |

The results show that one of the objectives has given a negative result in its review (4.1.e, safety related). This is justified because the project itself indicates that it is not covered. It is therefore an expected result.

## VI. CONCLUSION

This paper presents an ES based on rules that facilitates the work of checking compliance to standards DO-178C and DO-278A for development projects in the area of critical systems. This system allows users with a little experience in the field of standards and process management to check compliance problems, which normally would require a highly qualified and experienced specialist, while also enabling these non-expert users to improve their skills in the area.

Following the proposed methodology, the problem has been identified as the difficulty of using specific standards in the software development of complex system focused or airborne and ground systems. It is suggested that ES represent a very convenient method to facilitate this work as we consulted literature, which frames the relevance of ES in the context of Critical Systems, applicable in several varied domains. Due to the fact that the standards to be taken as a basis for this ES do not define a specific life cycle, we studied other similar standards with some common aspects, such as the use of critical levels, compliance with requirements, objectives, activities, etc. All of them are somehow based on the use of the processes. Once the common parts of these standards have been analysed, we introduced the concepts to model a life cycle. We developed an ES of generic application, which allows the evaluation of different life cycles for different standards, which configuration is made through some configuration files to customize its behaviour. The ES has been customized for the specific use of the standard DO-178C and DO-278A, thanks to the adaptation through this type of configuration files.

We proposed a method to evaluate software development processes based on the standards DO-178C and DO-278A, without depending on a specific life cycle. This method has generated process units, which interact with each other and can be adapted to any life cycle. There is a proposed framework where all the processes, outputs, inputs and dependencies between them are identified.

Results collected during validation confirm the perception of users that our ES facilitates the use and the associated learning of these standards. The compliance check work made by a technical user with the ES and by an expert without the ES gave similar results (once fixed one error found out during the process) but getting outcome was faster and required less effort with the ES. This happens because the user needs to identify the objectives, activities, items and selection of a set of checklists when using the manual method, with a considerable investment of time; with the ES, this is automatic and immediate. This increment of time and effort is higher in the manual method if the user is not an expert, while there is no difference with the automatic procedure of the ES.

The opinion of users expressed through the questionnaire clearly confirmed that they perceived ES as efficient and satisfactory. Also, usability is confirmed by users and they consider the ES as a helpful tool for the evaluation of standard compliance in development projects. All these data suggest that our ES is useful for facilitating the control, management and verification of development processes based on these standards without the assistance of one of the scarce experts in the area thus helping to solve the problems, which inspire our work.

Our research focuses on modelling all processes related to the standard under study (DO-178C and DO-278A), but

no limiting the future use in other standards. We propose a method of knowledge extraction, which could be valid for other similar standards. And finally, we have developed an intelligent resource (ES) that can be used not only by these standards but other similar ones through minimal configuration. This is one of the main contributions of this research.

One important point is that the results produced by this ES should not be taken as a final decision since all processes and evidences associated to compliance to these standards shall be accepted by the Approval or Certification Authority. This process facilitates generating all the compliance evidences to verify that the system is certifiable/approvable. Subsequently, the corresponding Authority will certify/approve the system through a Certification/Approval process. The Authority will decide if it is necessary to add more evidences, to add more objectives or feedback to the system with direct observation referred to the provided evidences.

## VII. FUTURE WORKS

The connection of the ES to the opportunity of also serving to help users to learn more on the standards and the processes is one promising area of work we are already working on: e.g. incorporation of Intelligent Tutoring System (ITS) learning modules, which would allow the user to access real-time tutoring functionality.

However, there is still obvious room for improvement of the system. One is the graphical user interface, which could be improved in new versions for better user experience. As the generation of this specific ES is based on a generic ES structure, it is also possible to adapt to new standards in a fast and efficient way through configuration files. Although the scope of this work is limited to the application of specific standards, its flexibility allows further adaptation to other standards, which work with the same structure: criticality levels, evaluation by processes, etc. For example, the ES can be improved with the addition of supplements like DO-330, DO-331, DO332 and DO-333. Its area of application is more specific and depend on the development techniques applied to the project, but we are already working in this addition to complete it in the near future.

## APPENDIX

This section includes two use cases as examples in the validation of the ES. Next, we show the checklists for each Data Item that the ES proposes for evaluation of compliance.

*Software Requirements Standards:*

For this data item, the following objectives and activities have been checked:

- **DO-178C - 4.1.e:** Are the Software Requirements Standards defined?
  According to Appendix B, Section B.2, page B-5: Yes, the Software Requirements Standards are defined.
- **DO-178C - 4.1.e:** Are the Software Requirements Standards consistent with the system's safety objectives?
  According to Appendix B, Section B.2, page B-5: No, for the GCS project there is no system safety assessment, therefore this objective cannot be checked.
- **DO-178C - 4.5.a** : Does the Software Requirements Standards meet the definition in section 11.6?
  Yes, this activity has been checked in the checklists.
- **DO-178C - 4.5.b:** Does the Software Requirements Standards allow the Software requirements to be uniformly defined?
  According to Appendix B, Section B.2.2, page B-7: Yes, these standards allow define the Software requirements. They are precise, and verifiable as possible.
- **DO-178C - 4.5.c:** Does the Software Requirements Standards prevent the use of constructions or methods that may produce outputs that cannot be verified or are not compatible with safety-related requirements (safety)?
  According to Appendix B, Section B.2, page B-5: No, there is no system safety assessment, therefore this activity cannot be.
- **DO-178C - 11.6.a:** Does the Software Requirements Standards include the methods to be used to develop Software requirements, such as structured methods?
  According to Appendix B, Section B.2, page B-5: Yes, Engineers used a version of the structured analysis for real-time system specification methodology. In general, the structured analysis method is based on a hierarchical approach to defining functional modules and the associated data and control flows.
- **DO-178C - 11.6.b:** Does the Software Requirements Standards include notations to be used to express needs, such as data flow diagrams and formal specification languages?
  According to Appendix B, Section B.2.1, page B-6: Yes, the specification document includes data context and flow diagrams, control and context flow diagrams, and process and control descriptions.
- **DO-178C - 11.6.c:** Does the Software Requirements Standards include limitations on the use of tools to develop requirements?
  According to Appendix B, Section B.2.1, page B-6: Yes, no constraints were placed on the use of requirements development tools.
- **DO-178C -11.6.f:** Does the Software Requirements Standards include the methods to be used to provide derived requirements from system processes?
  According to Appendix B, Section B.2.3, page B-8: Yes, this section includes some specific methods to provided Derived requirements.
  *Software Design Standards:*
  For this data item, the following Objectives and activities have been checked:
- **DO-178C - 4.1.e:** Are the Software Design Standards defined?
  According to Appendix B, Section B.3, page B: Yes, the design of a GCS implementation should be developed using the structured analysis and design methods.
- **DO-178C - 4.1.e:** Is the Software Design Standards consistent with the system's safety objectives?

According to Appendix B, Section B.2, page B-5: Yes, the Software Requirements Standards are defined.

- **DO-178C - 4.5.a:** Does the Software Design Standards meet the definition in section 11.7?
  Yes, this activity has been checked in the checklists.
- **DO-178C - 4.5.b:** Does the Software Design Standards allow requirements to be uniformly defined?
  According to Appendix B, Section B.3, page B-8: Yes, these Standards enable uniform design in the software implementations.
- **DO-178C - 4.5.c**: Does the Software Design Standards avoid the use of constructions or methods that may produce outputs that cannot be verified or are not compatible with safety requirements?
  According to Appendix B, Section B.2, page B-5: No, there is no system safety assessment, therefore this activity cannot be checked.
- **DO-178C - 11.7.a:** Does the Software Design Standards include a description of the design methods to be used?
  According to Appendix B, Section B.3.1, page B-9: Yes; the design of a GCS implementation should be developed using the structured analysis and design methods.
- **DO-178C - 11.7.b:** Does the Software Design Standards include a naming convention to be used?
  According to Appendix B, Section B.3.1, page B-10: Yes, In addition, no other design standards have been defined regarding naming conventions, scheduling, global data, or exception handling beyond those requirements set forth in the GCS specification.
- **DO-178C - 11.7.c:** Does the Software Design Standards include conditions imposed on permitted design methods?
  According to Appendix B, Section B.3.1, page B −10 Yes, Although no constraints in terms of project standards have been placed on the use of event-driven architectures, dynamic tasking, and re-entry, the use of such methods in a GCS design should be discussed and the rationale for their use clearly explained in the design documentation. Further, no formal constraints have been placed on the use of recursion, dynamic objects, data aliases and compacted expressions.
- **DO-178C - 11.7.d:** Does the Software Design Standards include limitations on the use of design tools?
  According to Appendix B, Section B.3.1, page B-9: Yes, the designer is required to use the Computer Aided Software Engineering (CASE) tool (Teamwork). In this section a set of limitations are included.
- **DO-178C - 11.7.e:** Does the Software Design Standards include design limitations?
  According to Appendix B, Section B.3.1, page B-10: Yes, Further, no formal constraints have been placed on the use of recursion, dynamic objects, data aliases and compacted expressions. However, as stated above, the use of such techniques should be clearly discussed and justified in the design documentation.
- **DO-178C - 11.7.f:** Does the Software Design Standards include complexity restrictions?
  According to Appendix B, Section B.3.1, page B-9: Yes, No restrictions have been issued on the complexity of the design, such as limiting the number of nested calls or entry and exit points in the code components.

## REFERENCES

[1] I. Sommerville, *Software Engineering*. Reading, MA, USA: Addison-Wesley, 2007, pp. 1–23.
[2] *Software Considerations in Airborne Systems and Equipment Certification*, document RTCA, DO-178C, 2012.
[3] *Software Integrity Assurance Considerations for Communication, Navigation*, document RTCA, DO-278A, Surveillance and Air Traffic Management (CNS/ATM), 2012.
[4] S. Jacklin, "Certification of safety-critical software under DO-178C and DO-278A," in *Infotech Aerospace* (Infotech Aerospace Conferences). American Institute of Aeronautics and Astronautics, Jun. 2012. [Online]. Available: http://dx.doi.org/10.2514/6.2012-2473
[5] S. Nair, J. L. de la Vara, M. Sabetzadeh, and L. Briand, "An extended systematic literature review on provision of evidence for safety certification," *Inf. Softw. Technol.*, vol. 56, no. 7, pp. 689–717, Jul. 2014.
[6] S. Nair, J. L. de la Vara, M. Sabetzadeh, and D. Falessi, "Evidence management for compliance of critical systems with safety standards: A survey on the state of practice," *Inf. Softw. Technol.*, vol. 60, pp. 1–15, Apr. 2015.
[7] M. A. A. Oroumieh, S. M. Bagher Malaek, M. Ashrafizaadeh, and S. M. Taheri, "Aircraft design cycle time reduction using artificial intelligence," *Aerosp. Sci. Technol.*, vol. 26, no. 1, pp. 244–258, Apr. 2013.
[8] E. Castillo, J. M. Gutiérrez, and A. S. Hadi, *Sistemas Expertos Y Modelos de Redes Probabilísticas*. Marqués de Salamanca Palace, Madrid: Academia de Ingeniería, 1997.
[9] J. Palma and R. Marín, *Inteligencia Artificial, Técnicas, Métodos y Aplicaciones*. New York, NY, USA: McGraw-Hill, 2008.
[10] X. Ge, R. F. Paige, and J. A. McDermid, "An iterative approach for development of safety-critical software and safety arguments," in *Proc. Agile Conf.*, Aug. 2010, pp. 35–43.
[11] S. Subair, "The evolution of software process models: From the waterfall model to the Unified Modelling Language (UML)," *Int. J. Inf. Technol. Syst.*, vol. 3, no. 2, pp. 7–14, 2014.
[12] A. B. Sasankar and V. Chavan, "Survey of software life cycle models by various documented standards," *Int. J. Comput. Sci. Technol.*, vol. 2, no. 4, pp. 137–144, 2011.
[13] *Guidelines for Development of Civil Aircraft and Systems*, document SAE ARP-4754A, Dec. 2010.
[14] *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*, document SAE ARP-4761, Dec. 1996.
[15] *Guidelines for ANS Software Safety Assurance*, document EUROCAE ED-153, 2012.
[16] *Fundamental Safety of Electrical/Electronic/Programmable Electronic Safety Related Systems*, document IEC-61508, International Electrotechnical Commission, 2010.
[17] *Design Assurance Guidance for Airborne Electronic Hardware*, document RTCA DO-254, Apr. 2000.
[18] *Road Vehicles—Functional safety*, document ISO-26262, 2018.
[19] *Functional Safety—Safety Instrumented Systems for the Nuclear Industries*, document IEC-61513, 2011.
[20] *Nuclear Power Plants—Instrumentation and Control Important to Safety—Classification of Instrumentation and Control Functions*, document IEC-61226, 2009.
[21] *Railway Applications—the Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)*, document EN-50126, 2005.
[22] *Railway Applications—Communications, Signalling and Processing Systems—Software for Railway Control and Protection Systems*, document EN-50128, 2012.
[23] *Railway Applications—Communications, Signalling and Processing Systems—Safety Related Electronic Systems for Signalling*, document EN-50129, 2003.
[24] *Medical Electrical Equipment*, document IEC-60601, 2012.

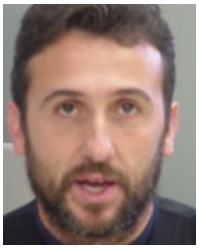
[25] *Medical Device Software—Software Life Cycle Processes*, document IEC-62304, 2006.
[26] B. W. Boehm, "Verifying and validating software requirements and design specifications," *IEEE Softw.*, vol. 1, no. 1, pp. 75–88, Jan. 1984.
[27] O. Laitenberger and J.-M. DeBaud, "An encompassing life cycle centric survey of software inspection," *J. Syst. Softw.*, vol. 50, no. 1, pp. 5–31, Jan. 2000.
[28] *National Research Council. Artificial Intelligence Applications of Critical Transportation Issues*, Transportation Research Board, Washington, DC, USA, 2012.
[29] D. J. Lary, "Artificial intelligence in aerospace," in *Aerospace Technologies Advancements*. Rijeka, Croatia: InTech, 2010.
[30] M. V. de Oliveira and J. C. S. de Almeida, "Application of artificial intelligence techniques in modelling and control of a nuclear power plant pressurizer system," *Prog. Nucl. Energy*, vol. 63, pp. 8–71, Mar. 2013.
[31] A. Agah, *Medical Applications of Artificial Intelligence*. Boca Raton, FL, USA: CRC Press, 2013.
[32] R. Varadaraju. *A Survey of Introducing Artificial Intelligence Into the Safety Critical System Software Design Process*. Accessed: 2011. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.114.8602&rep=rep1&type=pdf
[33] S. Karmore and D. A. R. Mahajan, "Testing of various embedded system with artificial intelligence approach," *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 4, no. 3, pp. 809–814, Mar. 2015. [Online]. Available: http://ijarcet.org/wp-content/uploads/IJARCET-VOL-4-ISSUE-3-809-814.pdf
[34] X. Zhu and S. Shen, "Application of fault diagnosis expert system for aircraft electric power system," in *Proc. ICAS*, 2002, pp. 1–4.
[35] E. Bechhoefer, A. Fang, and D. Van Ness, "Improved rotor track and balance performance using an expert system," in *Proc. IEEE Conf. Prognostics Health Manage.*, Jun. 2011, pp. 1–8.
[36] F. M. Rachel and P. S. Cugnasca, "Using artificial intelligence techniques in critical safety systems," in *Proc. 1st Inst. Eng. Technol. Int. Conf. Syst. Saf.*, London, U.K., 2006, pp. 64–70.
[37] A. Hauge and A. Tonnesen, "Use of articial neural networks in safety critical systems," M.S. thesis, Dept. Comput. Sci., 2004, pp. 1–93.
[38] S. Goss and G. Murray, *Artificial Intelligence Applications in Aircraft Systems*, document No. DSTO-RR-0071, Defence Science And Technology Organization, Canberra, ACT, Australia, 1996.
[39] L. Harrison, P. Saunders, and J. Janowitz, *Artificial Intelligence with Applications for Aircraft*. Pleasantville, NJ, USA: Galaxy Scientific Corporation, 1994.
[40] *FAA-Advisory Circular*, document 20-115D, 2017.
[41] *Recognition of ED-12C/DO-178C in EASA AMC 20-115 (Software Considerations for Airborne Systems and Equipment Certification)*, Eur. Aviation Saf. Agency, Cologne, Germany, 2013.
[42] J. Giarratano and G. Riley, *Expert Systems: Principles and Programming*, vol. 632. Boston, MA, USA: PWS-KENT Publishing Co, 1998.
[43] C. Niemann-Delius, *Proceedings of the 12th International Symposium Continuous Surface Mining-Aachen 2014*. New York, NY, USA: Springer, 2014.
[44] R. Moore, "Expert systems for process control," *TAPPI J.*, vol. 6, pp. 64–67, 1985.
[45] J. P. Gipe and N. D. Jasinski, "Expert system applications in quality assurance," in *ASQ Quality Congress Transactions*, vol. 40. Anaheim, CA, USA: American Society for Quality Control, 1986, pp. 280–284.
[46] E. P. Paladini, "An expert system approach to quality control," *Expert Syst. Appl.*, vol. 18, no. 2, pp. 133–151, Feb. 2000.
[47] H.-T. Liao, D. Enke, and H. Wiebe, "An expert advisory system for the ISO 9001 quality system," *Expert Syst. Appl.*, vol. 27, no. 2, pp. 313–322, Aug. 2004.
[48] K. Eldrandaly, "A knowledge-based advisory system for software quality assurance," *Int. Arab J. Inf. Technol.*, vol. 5, no. 3, pp. 1–7, 2008.
[49] M. Liukkonen, E. Havia, H. Leinonen, and Y. Hiltunen, "Expert system for analysis of quality in production of electronics," *Expert Syst. Appl.*, vol. 38, no. 7, pp. 8724–8729, Jul. 2011.
[50] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?" *Int. J. Hum.-Comput. Stud.*, vol. 43, nos. 5–6, pp. 907–928, Nov. 1995.
[51] D. R. Hernández, J. V. Labrada, and M. R. Hernández, "Ontologías. Integración de Esquemas," *Tlatemoani*, vol. 17, pp. 16–32, 2014.
[52] V. P. Ramavath and A. S. Moharir, "An Approach to Semantic Search using Domain Ontology and GraphDB," *Int. J. Appl. Eng. Res.*, vol. 13, no. 15, pp. 11707–11714, 2018.
[53] S. R. Chaudhuri, A. Banerjee, N. Swaminathan, V. Choppella, A. Pal, and P. Balamurali, "A knowledge centric approach to conceptualizing robotic solutionse," in *Proc. 12th Innov. Softw. Eng. Conf. (Formerly Known India Softw. Eng. Conf.)*, 2019, pp. 1–11.
[54] T. H. Al Balushi, P. R. F. Sampaio, D. Dabhi, and P. Loucopoulos, "ElicitO: A quality ontology-guided NFR elicitation tool," in *Proc. Int. Working Conf. Requirements Eng., Found. Softw. Quality*, Berlin, Germany: Springer, 2007, pp. 306–319.
[55] E. Mezghani, E. Exposito, and K. Drira, "A collaborative methodology for tacit knowledge management: Application to scientific research," *Future Gener. Comput. Syst.*, vol. 54, pp. 450–455, Jan. 2016.
[56] M. Fernández, A. Gómez-Pérez, and N. Juristo, "Methontology: From ontological art towards ontological engineering," in *Proc. AAAI Symp.*, Stanford, CA, USA: Univ. Stanford, 1997, pp. 33–40.
[57] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *J. Manage. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, Dec. 2007.
[58] W. Kuechler and V. Vaishnavi, "The emergence of design research in information systems in North America," *J. Design Res.*, vol. 7, no. 1, pp. 1–16, 2008.
[59] R. Winter, "Design science research in europe," *Eur. J. Inf. Syst.*, vol. 17, no. 5, pp. 470–475, Oct. 2008.
[60] A. R. Hevner and S. Chatterjee, "Design science research in information systems," in *Association for Information Systems*, J. vom Brocke, Ed. New York, NY, USA: Reference Syllabi, 2015. Accessed: Jul. 27, 2018. [Online]. Available: http://eduglopedia.org/reference-syllabus/AIS_Reference_Syllabus_Design_Science_Research_in_IS.pdf
[61] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design research in information systems research," *MIS Q*, vol. 28, no. 1, pp. 75–105, 2004.
[62] M. De Vries, "A method to enhance existing business-it alignment approaches," *South African J. Ind. Eng.*, vol. 24, no. 2, pp. 11–126, 2013.
[63] B. Kuechler and V. Vaishnavi, "On theory development in design science research: Anatomy of a research project," *Eur. J. Inf. Syst.*, vol. 17, no. 5, pp. 489–504, Oct. 2008.
[64] A. K. Carstensen and J. Bernhard, "Design science research as an approach for engineering education research," in *Proc. 12th Int. CDIO Conf. Turku Univ. Appl. Sci.*, 2016, pp. 1072–1081.
[65] S. Gregor and A. R. Hevner, "Positioning and presenting design science research for maximum impact," *MIS Quart.*, vol. 37, no. 2, pp. 337–355, Feb. 2013.
[66] H. Takeda, P. Veerkamp, and H. Yoshikawa, "Modeling design process," *AI Mag.*, vol. 11, no. 4, p. 37, 1990.
[67] V. K. Vaishnavi and W. Kuechler, *Design Science Research Methods and Patterns: Innovating Information and Communication Technology*. Boca Raton, FL, USA: CRC Press, 2015.
[68] J. Pries-Heje and J. Venable, "Evaluation risks in design science research: A framework," in *Int. Conf. Design Sci. Res. IT (DESRIST)*, lanta, Georgia: Robinson College of Business, Georgia State University, 2008, pp. 329–334.
[69] M. K. Sein, O. Henfridsson, S. Purao, M. Rossi, and R. Lindgren, "Action design research," *MIS Quart.*, vol. 35, no. 1, pp. 37–55, Mar. 2011.
[70] S. T. March and G. F. Smith, "Design and natural science research on information technology," *Decis. Support Syst.*, vol. 15, no. 4, pp. 251–266, Dec. 1995.
[71] A. Hevner and S. Chatterjee, *Design Research in Information Systems: Theory and Practice*, vol. 22. New York, NY, USA: Springer, 2010.
[72] W. Kuechler, U. of Nevada, V. Vaishnavi, and G. State University, "A framework for theory development in design science research: Multiple perspectives," *J. Assoc. Inf. Syst.*, vol. 13, no. 6, pp. 395–423, Jun. 2012.
[73] V. Vaishnavi and W. Kuechler. *Design Research in Information Systems*. Accessed: Dec. 11, 2018. [Online]. Available: http://desrist.org/design-research-in-information-systems
[74] J. A. Jiménez, J. A. M. Merodio, and L. F. Sanz, "Checklists for compliance to DO-178C and DO-278A standards," *Comput. Standards Interfaces*, vol. 52, pp. 41–50, May 2017.
[75] *RTCA DO-178B Process Visual Summary*. Accessed: Dec. 11, 2018. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/4/4f/DO-178B_Process_Visual_Summary_Rev_A.pdf
[76] L. Rierson, *Developing safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance*. Boca Raton, FL, USA: CRC Press, 2013.
[77] V. Hilderman and T. Baghai, *Avionics Certification: A Complete Guide to DO-178 (Software), DO-254 (Hardware)*. Leesburg, VA USA: Avionics Communications, Inc., 2007.
[78] *JRuleEngine*. Accessed: Dec. 11, 2018. [Online]. Available: http://jruleengine.sourceforge.net

[79] S. Walli, D. Gynn, and B. Von Rotz, "The growth of open source software in organizations," *Report*, pp. 1–18, 2005. [Online]. Available: https://www.immagic.com/eLibrary/ARCHIVES/GENERAL/OPTAR_US/O051219W.pdf
[80] V. Nehra and A. Tyagi, "Free open source software in electronics engineering education: A survey," *Int. J. Modern Edu. Comput. Sci.*, vol. 6, no. 5, pp. 15–25, May 2014.
[81] A. J. Gonzalez and D. D. Dankel, *The Engineering of Knowledge-Based Systems*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1993, pp. 207–230.
[82] C. Y. Suen, P. D. Grogono, R. Shinghal, and F. Coallier, "Verifying, validating, and measuring the performance of expert systems," *Expert Syst. Appl.*, vol. 1, no. 2, pp. 93–102, Jan. 1990.
[83] M. Suwa, A. C. Scott, and E. H. Shortliffe, "An approach to verifying completeness and consistency in rule-based expert systems," *AI Mag.*, vol. 3, no. 2, pp. 16–21, 1982.
[84] C. Huang and M. Cheng, "Conflicting treatment model for certainty rule-based knowledge," *Expert Syst. Appl.*, vol. 35, nos. 1–2, pp. 161–176, Jul. 2008.
[85] T. A. Nguyen, W. A. Perkins, T. J. Laffey, and D. Pecora, "Knowledge-base verification," *AI Mag.*, vol. 8, no. 2, pp. 69–75, 1987.
[86] *Radio Technical Commission for Aeronautics (RTCA), European Organization for Civil Aviation Electronics (EUROCAE)*, document no. DO-178B/ED-12B, RTCA: Software Considerations in Airbone Systems and Equipment Certification, Dec. 1992.
[87] K. J. Hayhurst, *Guidance and Control Software Project Data: Planning Documents*, vol. 1. Hampton, VA, USA: NASA Langley Research Center, 2008.
[88] K. J. Hayhurst, *Guidance and Control Software Project Data: Development Documents*, vol. 2. Hampton, VA, USA: NASA Langley Research Center, 2008.
[89] K. J. Hayhurst, *Guidance and Control Software Project Data: Verification Documents*, vol. 3. Hampton, VA, USA: NASA Langley Research Center, 2008.
[90] K. J. Hayhurst, *Guidance and Control Software Project Data: Configuration Management and Quality Assurance Documents*, vol. 4. Hampton, VA, USA: NASA Langley Research Center, 2008.
[91] S. Godara and R. Singh, "Evaluation of predictive machine learning techniques as expert systems in medical diagnosis," *Indian J. Sci. Technol.*, vol. 9, no. 10, pp. 1–14, Mar. 2016, doi: 10.17485/ijst/2016/v9i10/87212.
[92] R. Likert, "A technique for the measurement of attitudes," *Arch. Psychol.*, vol. 22, pp. 1–55, 1932.

**JOSE ANDRES-JIMENEZ** received the degree in electronic engineering, in 2012. He received the master's degree in advanced electronic systems and intelligent systems, in 2014, and the technical engineering degree in telecommunications, specializing in electronic systems from Alcalá University. He worked as a Computer Consultant with Alcalá University for six years. He is currently working in Indra as a Safety Manager from different projects and areas. He has experience in the Safety Assessment of Critical Systems and in the Software Assurance in different applications and environments (Railway, Naval, Aerospace, Airborne, and Radar Systems). He is a Member Airworthiness office, where he works as a Safety Compliance Verification Engineer (CVE). His research interests include safety and risk assessment, in the transport sector. Other area of interest is focused in artificial intelligence. He is working in his Ph.D. in Information and Knowledge Engineering focused in the application of the Artificial Intelligence in the development of standards for Safety Critical Systems.

**JOSE-AMELIO MEDINA-MERODIO** graduated in marketing and market research from Rey Juan Carlos University. He received the degree in telecommunications engineering from the University of Alcalá, and the Official Master's degree in company management and the Ph.D. degree in administration and business management from Rey Juan Carlos University. He is currently an Assistant Professor with the Department of Computer Science, University of Alcalá. He acted as a Technical Management (Systems and Information Technology) member with the Health Service of Castille-Mancha, as a Computer Technician with the ADAC Association, and as the Head of quality, environment and computer services with Tecnivial, SA Company.

**LUIS FERNANDEZ-SANZ** received the degree in computing from the Polytechnic University of Madrid (UPM), in 1989, and the Ph.D. degree in computing with a special award at the University of the Basque Country, in 1997. He is currently an Associate Professor with the Department of Computer Science, University of Alcalá (UAH). With more than 20 years of research and teaching experience (at UPM, Universidad Europea de Madrid, and UAH), he has also been involved in the management of the main Spanish Computing Professionals association (ATI) as the Vice President. He is also the Chairman of ATI Software Quality group. He has held the position of Vice President of Council of European Professional Informatics Societies (CEPIS), from 2011 to 2013 and since 2016. His general research interests include software quality and engineering, accessibility, and ICT professionalism and education.



**JOSE-JAVIER MARTINEZ-HERRAIZ** received the University degree in computer science from the PolyTechnic University of Madrid, and the Ph.D. degree from Alcalá University, in 2004. He has worked in private telecommunication business companies (Spain and Italy), such as an Analyst, the Project Manager, and a Consultant, from 1988 to 1999, and, at present, has been a Professor with the Department of Computer Science (Artificial Intelligence Area), University of Alcalá, since 1994. From 2004 to 2006, he was the Deputy Director with the Higher Technical School of Computer Engineering—Universidad de Alcalá. From 2006 to 2012, he was an Educational Partner with Lund University, Sweden. From 2008 to 2011, he was the Director of the Department of Computer Science. From 2010 to 2015, he was the Director Amaranto-UAH Research Chair of Digital Security and Internet of Future. From 2015 to 2018, he was the Co-Director DARS-UAH Research Chair DARS-UAH of Cyberintelligence. Since 2017, he has also been the Co-Director ISDEFE-UAH Research Chair of Cybersecurity. He has practical work experience on software process technology and modeling, methodologies to software projects planning and managing, software maintenance, e_learning gamification technology, and cybersecurity Projects. He has collaborated extensively with Spanish Law Enforcement Agencies (Cuerpo Nacional de Policía, Guardia Civil y Fiscalía de Criminalidad Informática) and cybersecurity companies (Ingeniería de Sistemas de la Defensa—ISDEFE—Ministerio de Defensa, Prosegur Ciberseguridad, Eulen Seguridad). He received the award in the Order of Merit of the Police Forces (Guardia Civil), in 2011, and the Order of Merit of the Police Forces (Cuerpo Nacional de Policía), in 2015. Cross of Merit of Security and Protection San Cristóbal de Magallanes (International Association of Security and Civil Protection), in 2019.

**JAVIER GONZALEZ-DE-LOPE** received the computing engineering degree from the University of Alcalá (UAH), in 2017. In 2016, he started his career as a Full Stack Developer of web applications at Visiotech, where he is currently the Lead Developer of a cloud-based project focused on video surveillance device monitoring. His main areas of interest are cloud computing design, software quality assurance, container orchestrated application development, and JavaScript based technologies.