Empirical Software Engineering (2023) 28:100
https://doi.org/10.1007/s10664-023-10350-7

Improving hardware/software interface management in systems of systems through documentation as code

Héctor Cadavid[1,2] · **Vasilios Andrikopoulos**[1] · **Paris Avgeriou**[1]

Accepted: 30 May 2023 / Published online: 6 July 2023

Abstract

Context The management of *Interface Control Documents* (ICDs) has shown to be a major pain point in the architecting processes of *Systems of Systems* (SoS).

Objective This work aims to improve on previously identified ICD management issues using the documentation-as-code philosophy as a potential basis for a treatment, and in collaboration with practitioners.

Method We conducted a Technical Action Research (TAR) study with a group of engineers at the Netherlands Radio Astronomy Institute (ASTRON), in the context of the LOFAR radio telescope. An additional research instrument, in the form of an expert panel, was used to evaluate the transferability of the proposed treatment to alternative domains.

Results In-depth insights on previously identified interface management issues were gained. Based on these insights a functional proof-of-concept was developed aimed at addressing these issues following the documentation-as-code principles. In addition to receiving overall positive reviews from practitioners and experts, further areas of improvement and transferability considerations for future work were identified.

Conclusions The proposed approach, which to our knowledge has not been explored before in this context, is promising to address some of the recurring interfacing-related issues with directed SoS in multiple engineering domains. This could be done mainly by enforcing consistency and completeness on both text-based and formal elements of the ICDs, and turning ICDs into single sources of truth for the architecting processes of large scale SoS.

Communicated by: Hèlène Waeselynck

✉ Héctor Cadavid
h.f.cadavid.rengifo@rug.nl

Vasilios Andrikopoulos
v.andrikopoulos@rug.nl

Paris Avgeriou
p.avgeriou@rug.nl

[1] Department of Computer Science, Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, Broerstraat 5, Groningen 9712 CP, Groningen, Netherlands

[2] Department of Informatics, Universidad Escuela Colombiana de Ingeniería, Ak 45#205-59, Bogotá 111166, Cundinamarca, Colombia

Keywords System of systems · Interface control document · Documentation as code · Technical action research

1 Introduction

A System-of-Systems (SoS) is by definition (Maier 1998; ISO/IEC 2019) *a family of independent systems that cooperate with each other to fulfill a common goal or provide capabilities that each system cannot achieve individually*. This kind of complex, large-scale systems is particularly present in domains such as defense, space, aeronautics, and energy. Given the strategic value of this kind of systems in these domains, a significant body of knowledge on the SoS architecting processes has accummulated in the past three decades (Cadavid et al. 2020).

However, a series of recent studies have identified a particular aspect of these processes that, despite having a significant contribution to major integration and operational issues, has not been adequately explored (Cadavid et al. 2020): the interplay between the disciplines involved in the architecting. This aspect was highlighted in particular through case studies in the radio-astronomy domain (Cadavid 2021; Cadavid et al. 2022), where, among others, the management of interfaces — the element where the various disciplines meet — was identified as a major pain point in SoS architecting. Particularly, these studies suggest that the details of these interface specifications, referred to as *Interface Control Documents* or *ICDs* in the systems engineering jargon, when defined by one of the involved disciplines, are often incomplete or not sufficiently clear for the other disciplines to work with. This, in addition to the knowledge gap between the involved disciplines, appears to be a common cause of *misunderstandings* and *wrongful assumptions* in the process (Sheard et al. 2018; Cadavid et al. 2020).

With this motivation, the authors, in cooperation with a group of engineers of the *Netherlands Institute for Radio Astronomy* (ASTRON)[1], conducted a *technical action research* (TAR) study aimed at exploring alternatives to address these issues (Cadavid et al. 2022). In particular, the treatment designed and evaluated through this TAR was tailored to the aforementioned issues in the context of ICD management of hardware/software interfaces in LOFAR, a large-scale, long-running, directed SoS designed and maintained by ASTRON. Moreover, the *documentation-as-code* or *docs-as-code (DaC)* philosophy (Gentle 2017), which encourages the creation and maintenance of technical documentation as rigorously as software, was used as the basis for said treatment. There were two reasons for that. First, the researchers identified that similar problems of unreliable documentation, in the context of software systems, have been partially addressed in the past with DaC pipelines[2]. Second, ASTRON practitioners had already been experimenting with a limited, ad-hoc version of a DaC approach to overcome the limitations of human-only readable ICDs (e.g., automating the generation of error-prone artifacts through an intermediate, machine-readable version of the ICD's technical specifications).

The contributions of this TAR study can therefore be summarized as follows. First, more in-depth insights on the ICD-management related issues as identified in previous studies (Cadavid et al. 2020; Cadavid 2021; Cadavid et al. 2022) were gathered. Second, an adaptation of the DaC philosophy, in terms of the features it should provide, is presented

[1] https://www.astron.nl/

[2] A collection of conferences, video casts and articles from practitioners that support this statement is available at https://www.writethedocs.org/guide/docs-as-code/

as an alternative for managing ICDs in document-centered engineering process, particularly in SoS. Third, a functional proof-of-concept of these DaC features is implemented, including a software baseline for some of the features not supported by a regular DaC pipeline (e.g., hardware-oriented extensions for markup languages, APIs for tracking dependencies, etc.). Fourth, evidence of the merit of the proposed approach, together with points for its improvement, is collected.

Given these encouraging outcomes and considering that these are specific to a single domain, this paper reports an extension to the aforementioned TAR, published at the 16th European Conference on Software Architecture (ECSA) (Cadavid et al. 2022). In this extended version of the paper, we explore (in addition to providing further details on the original TAR study) their external validity. In other words, we investigate to what extent the proposed treatment would address the ICD-management issues in application domains beyond radio-astronomy. To this end, an expert opinion approach was used. The results of the expert panel not only confirm the applicability of this adaptation of the DaC philosophy in diverese domains such as *aeronautics*, *space*, *ground transportation*, *energy* and *defense*, but also provide further considerations for its actual implementation. Furthermore, the opinions of ASTRON practitioners and domain experts, when analyzed side by side also provide valuable insights into the overall concept as an alternative for interface management where and when pure model-based engineering processes are not convenient or applicable.

The rest of this paper is structured as follows: Section 2 summarizes the background and the works related to this study. Section 3 presents the study design, and Section 4 answers the stated research questions. Section 5 discusses the results of the TAR study in light of the expert panel findings, while Section 6 elaborates on threats to validity and their mitigation. Section 7 concludes the study and discusses future work.

2 Background and Related Work

2.1 LOFAR as a System-of-Systems - System Overview

Radio telescopes are scientific astronomy instruments used to study the radio frequency portion of the electromagnetic spectrum emitted by astronomical objects (i.e., frequencies ranging from 9kHz up to 300GHz). They allow to capture information from astronomical objects very far away which can not be attained by traditional optical telescopes. However, unlike optical telescopes, radio telescopes require a much larger area to achieve a usable resolution.

LOFAR, one of the largest radio telescopes in the world, was created and is currently maintained by the Netherlands Institute for Radio Astronomy (ASTRON), one of the world's leading research institutions in the area. Like other large-scale radio telescopes, LOFAR is made out of a collection of geographically distributed, independently operated radio stations that sample, digitize and filter data. Despite the operational independence of the LOFAR stations, however, they are subordinated to a centrally-managed purpose, that is conducting large-scale space surveys. These stations, when working together, provide an effective collecting area equivalent to 300,000 square meters (Beck 2015). Once the stations are coordinated for this common goal, their collected data is processed at a central location and is subsequently made accessible for use by astronomers.

Given the above, and according to Firesmith's SoS profiling model (Firesmith 2010), LOFAR could be classified as an *ultra-large-scale*, *directed SoS* with a *high level of complex-*

ity, made out of *globally distributed*, *independently governed* and *operationally independent* systems. Furthermore, given the nearly 20 years spent on the architecting, design and development of its two phases (LOFAR and LOFAR2.0), this system could also be described as a long-running one. Consequently, all the lessons learned in following a document-centered approach during such a long time-frame, can provide valuable insights in ICDs management for systems of the scale and complexity of LOFAR.

2.2 *DaC*—Documentation-as-Code

In recent years the software engineering community has been steadily shifting from traditional documentation approaches using conventional word processors, wikis, or other collaborative editing system to the DaC philosophy (Gentle 2017). With DaC, where documentation is managed in the same way as source code in modern software projects, the community has been aiming at improving well-known issues such as outdated or unreliable technical documentation. Consequently, DaC implies the use of lightweight text-based markup languages (instead of proprietary formats and authoring tools) for the documents, so that they can be managed with modern (and proven) source code-oriented version control systems like Git and their related collaboration and automation tools. These automation possibilities mean, in turn, that a DaC documentation pipeline can ensure, to some extent, the quality of the documents by automatically validating or testing critical elements before their publication (e.g., the validity of code snippets within the documents, broken links, etc.). Furthermore, a DaC pipeline can also ensure uniformity and improve maintainability by using vendor-independent text-based specifications (e.g., for diagrams) that can be automatically transformed into visual elements that suit the organization's conventions.

Although the most common target of this documentation philosophy is software artifacts (e.g., APIs), it has also been adopted for higher-level documentation in industrial settings, showing to be useful to improve problems of missing or outdated documentation (Thomchick 2018) by reducing the complexity of documentation maintenance (Ozerova et al. 2020). Examples include the documentation of products in government systems (Lambourne 2017), architecture documentation in transport systems[3], and product engineering documentation[4]. Moreover, despite this topic having received little attention by researchers (Rong et al. 2020), it is worth highlighting the large community of technical writers working on it, with the near 2000 members of the DaC-global network in the *Write The Docs*[5] community as a prominent example.

2.3 *ICD Management Approaches*

In the context of Systems Engineering, an ICD is a formal description of an agreement for the interfacing between two or more systems. There are no conventions nor standards to define these artifacts, as they usually differ from one company to another (Rahmani and Thomson 2011), even within the same application domain (Louadah et al. 2014). However, when it comes to the approaches to manage these artifacts, it is fair to say that the existing ones fall somewhere between the two ends of the spectrum (Harvey et al. 2012): from pure model-centric, i.e., following a *model-based systems engineering process* (MBSE), to pure

[3] Deutsche Bahn - DB Systel - https://github.com/docToolchain/docToolchain

[4] OpenGADES (a work in progress) - https://wiki.eclipse.org/OpenADx

[5] https://www.writethedocs.org

document-centric ones. On the pure model-based end, the overall process is centered on a model of the system, from which documents like the ICDs are generated, when required, as a report. Examples in the context of SoS can be found in industries such as *astronomy* (Karban et al. 2018; Chiozzi et al. 2018), *space* (Di Maio et al. 2018; Vipavetz 2016), and *defense*(Tsui et al. 2018). In these cases, with the exception of the defense industry, which has adopted the *Unified Profile for DoDAF/MODAF* (UPDM), SysML is seemingly the de facto formalism in model-based approaches (Japs et al. 2021).

On the other extreme, the document-centric one, ICDs are mostly textual documents created with propietary word processing and diagramming tools, which evolve as interfaces are identified, defined, documented, and modified over time (Wheatcraft 2010). This approach, despite the growing popularity of model-based ones as a response to its limitations, is still widely used in industry (Broy et al. 2021). This is evidenced not only by examples in the literature in the *aeronautics* industry (Guo et al. 2020) and *radio-astronomy* (the LOFAR case) (van Haarlem et al. 2013), but in the many ICDs publicly available online for other domains[6]. Between these two extremes, there are approaches that neither follow an MBSE process nor use text-based documents for the ICDs. Instead, these make use of computer aided tools to model only the relationships between the parties involved in the described interface (rather than the overall system), so that computations (e.g., evaluating the impact of a change of a parameter) can be performed. Examples in the literature include the use of spreadsheets to model ICDs in *subsea production* (Yasseri and Bahai 2019) and *astronomy* systems (Borrowman and Taylor 2016); and UML for modelling the interfaces of *cyber-physical systems* Rahmani and Thomson (2011).

3 Study Design

In this section we discuss the goals of this work and the methods adopted to achieve these goals.

3.1 Study Goal and Research Questions

The goal of this study can be described following the *design problem* structure proposed by Wieringa Wieringa (2014), Chapter 19:

> **Improve** the ICD management practices
> **By** adapting the DaC philosophy to this context,
> **Such that** the managed ICDs lead to fewer erroneous assumptions and misunderstandings while engineers from different disciplines work with them,
> **In order to** reduce potential integration and operational issues caused by these occurrences.

Addressing this design problem involves identifying a set of documentation-management features (inspired by the DaC philosophy) that would deal with the issues that cause the said

[6] Particular examples are available in domains like https://www.hhs.gov/guidance/sites/default/files/hhs-guidance-documents/DDC_ICD_V020117_041019_v1_5CR_040919_RETIRED_2.pdf*Healthcare*, http://www.in2rail.eu/download.aspx?id=40938a15-24c4-427a-9e29-6811fabebaef*Transport systems*, http://www.h2020-ergo.eu/wp-content/uploads/ERGO_D1_3_InterfaceControlDocument_V2.2.pdf*Aerospace/robotics*, and https://eoepca.github.io/master-system-icd/current/#mainOIDC*Astronomy*

assumptions/misunderstandings. In addition to a general description of these issues, concrete instances of them, i.e., specific symptoms within actual ICDs, are necessary in order to elicit requirements of these features. Given this, the following research questions are proposed as the means to address the design problem:

- $RQ1$ What are the issues with ICDs management that cause erroneous assumptions and misunderstandings when working with these documents in SoS?
- $RQ2$ What are the symptoms of the previously identified ICD-management issues within a typical hardware/software-oriented ICD?
- $RQ3$ What are the features required for a DaC-based ICD management approach given the identified ICD elements and management issues?
- $RQ4$ What is the design of an ICD management pipeline that provides such features?
- $RQ5$ To what extent can the designed ICD management pipeline address the identified ICD management-related issues in the context of radio-astronomy scientific instruments?
- $RQ6$ To what extent can the proposed ICD management approach be transferred beyond the radio-astronomy domain, to other application domains?

As described in the introduction, the ICD-management related issues identified in preceding studies (Cadavid et al. 2020; Cadavid 2021) are among the main motivations for this study. However, as said studies had a broad scope, i.e. they looked at cross-disciplinary issues in general, the first research question (RQ1) is aimed at gaining more details on issues related to ICD-management. The second research question (RQ2) is aimed at identifying specific symptoms of the issues identified by RQ1 (i.e., what these issues look like in actual ICDs) as the base for eliciting requirements for the features that would address them. As hardware/software interfaces are among the most prominent (and critical) ones in systems like LOFAR, this question is focused on ICDs with this kind of interface definitions. The third research question, in turn, aims to define a set of features for an ICD-management pipeline that would prevent the identified issues to occur by addressing their related symptoms. Research questions four (RQ4) and five (RQ5), on the other hand, are aimed at creating a design for the proposed features and at evaluating its applicability in the LOFAR system, respectively. Finally, research question six (RQ6) seeks to evaluate to what extent the identified features could be transferred to domains beyond LOFAR's radio-astronomy.

3.2 Research Methods

The following elaborates on the two main research methods adopted in this study, namely *Technical Action Research* and *Expert Panel*, to address the six research questions previously described.

3.2.1 Technical Action Research

The first five research questions are addressed through the TAR method (Wieringa 2014, Chapter 19), as these require to investigate an experimental treatment (a DaC approach for ICDs management), to help a client with previously identified problems (ASTRON) and learn about its effects in practice. In particular, these questions were addressed through the phases of *diagnosing*, *action planning*, *action taking*, and *evaluation* of the TAR. In the last phase of the TAR, namely the *specifying learning* phase, the lessons learned from this study (which reflects on the first cycle of the action research), and their implications in future iterations of the study and the proposed documentation pipeline, were collected and are reported in the *Discussion* section of the paper.

The following elaborates on how the first four activities of the TAR addressed RQ1 to RQ5 and the participants involved in them.

Diagnosing
This phase of TAR is focused on exploring and extending the current understanding of ICD documentation problems identified in Cadavid (2021). Previous to this phase, the general idea of adopting DaC for ICD management was pitched to the potential participants. Subsequently, this phase used a *virtual focus group* with the ASTRON practitioners; a focus group is a qualitative research method to collect data on a given topic through a group interaction (Kontio et al. 2008). The discussion points for the focus group were derived from the relevant findings of the aforementioned previous study (Cadavid 2021). The actual focus group session was geared toward the collection of more context and details on ICD-management issues experienced in the LOFAR/LOFAR2.0 project in order to answer **RQ1**.

Action planning
This phase was focused on identifying, discussing, and choosing solutions to improve ICD-management related issues experienced by ASTRON practitioners. Exploring and discussing alternative solutions required, in the first place, understanding how the identified ICD-management related problems are manifested within an actual ICD, i.e., what are the symptoms of said problems (RQ2). As a guideline for ASTRON practitioners to properly identify said symptoms, and towards avoiding e.g., open-ended questions without enough context or exhibiting bias towards the particularities of a specific ICD, a generic ICD template with the most representative elements of a hardware/software oriented ICD was created. This template was annotated collaboratively by one of the researchers and a group of practitioners (see Table 1) to highlight specific symptoms of the issues identified in the diagnosing phase, e.g., particular elements of a given type of hardware elements that are often missed despite being key (**RQ2**). It is worth noting that this template was created by identifying the common elements between the ICDs provided by ASTRON engineers and a curated set of publicly available ICDs (available in the replication package[7]) that were (1) created under a document-centered process, and (2) focused on hardware-software interfacing descriptions. Finally, to address **RQ3**, researchers, in cooperation with one of the practitioners, defined a set of features that the DaC pipeline should provide in order to address the identified issues, using their particular issue symptoms to gauge their applicability in the process.

Action taking
Here the solutions selected to improve the identified issues are implemented. Therefore, in this phase the features identified in the previous phase were turned into an actual design that integrates existing DaC tools with any additional custom artifacts required by said features. This design is the answer to **RQ4**, and was implemented as a functional proof-of-concept that practitioners can try out and evaluate.

Evaluation
In this phase, the effects of the action are captured through different data collection methods (Petersen et al. 2014). This phase addressed **RQ5** by exploring the efficacy and fitness for purpose of the proposed documentation management approach to address the identified issues during the diagnosing phase. We chose to perform a *single case mechanism experiment*, an evaluation approach that involves testing the effectiveness of a treatment on a single case by examining the interaction between a designed artifact (such as a new product or

[7] https://figshare.com/s/671cc7450d8a6b8e2987

intervention) and the specific problem context that it is intended to address (Wieringa 2014, Chapter 18). That is to say, the experiment involves evaluating the impact of the treatment on a single individual or group, and examining the mechanisms through which it produces its effects. In our case, the experiment was conducted with the group of ASTRON engineers described in Table 1, and was used to evaluate the impact of the proposed ICD-mangement features on the issues identified in RQ1.

To this end, practitioners were asked to carry out a number of ICD management activities using an online instance of the functional proof-of-concept created in the action taking phase. In the process, one of the researchers provided help with the process through Slack. Once the activities have been performed, practitioners were asked to make an assessment of the applicability of the treatment to address the identified ICD-management related issues, and to ellaborate on it. This assessment, along with the observations or issues that emerged through the Slack interaction, were analyzed to answer RQ5 and followed through in the consequent *learning specification phase*. As the single-case experiment was tailored to the results of the first four research questions, the scenarios used for this evaluation will be further explored in the results section.

Participants and timeline

A total of ten people participated on the TAR part of the study, which took place between May 2021 and the beginning of May 2022: two researchers and eight ASTRON practitioners. One of the researchers actively participated in the design and development activities during the *action planning* and *action taking* phases; both researchers took part in the research activities, designing the research instruments, and drawing lessons from the *action evaluation* phase. The eight participants on the ASTRON side, on the other hand, were free to decide in which phase of the study they will participate. As a result, they participated in different phases of the study, as described in Table 1.

3.2.2 Expert Panel

The sixth research question (**RQ6**) was addressed through the *expert opinion* instrument (Wieringa 2014, Chapter 5). This instrument entails a panel of experts evaluating the design of an artifact by predicting the effects it would have, and how it would interact with potential problematic scenarios imagined by them. In our case, the functional description of the proposed features for the ICD management approach was used as the design to be evaluated. Moreover, the panel was organized by focusing the experts' predictions on the transferability of the TAR findings to other application domains.

In addition to ensuring that panel members have ample experience with ICDs and are thus fit-for-purpose for this panel, multiple resources were provided to them as the means to allow them to develop a proper understanding of this functional description. More specifically, in addition to making available our previous work (Cadavid et al. 2022), a series of introductory videos for the proposed features were produced; these videos included a series of screencasts of the functional prototype, using the features defined in RQ3 for creating and manipulating a fictional ICD that included hardware/software interfaces. Furthermore, the experts were given access to the ICD management platform developed in the TAR study to further explore the source documents and ICDs generated in the fictional case[8].

[8] Both introductory material and case sources are included in the replication package available at https://figshare.com/s/671cc7450d8a6b8e2987

Table 1 Practitioners involved in the different phases of the TAR study, and when such phases took place

ID	Role in LOFAR project	D	AP	AT	EV	SL
P1	Researcher	✓				
P2	Software Engineer	✓	✓		✓	
P3	RF Electronics Engineer	✓				✓
P4	Senior Software Engineer	✓	✓		✓	
P5	Head of Software Development	✓				
P6	Senior Software Engineer				✓	
P7	Software Engineer		✓		✓	
P8	System Engineer		✓		✓	
R1	N/A	✓	✓	✓	✓	✓
R2	N/A	✓			✓	✓
		May 2021- Jun 2021	Aug 2021- Jan 2022	Jan 2022- Mar 2022	Mar 2022- May 2022	Apr 2022- May 2022

P1, P2 ... P8: practitioners on the ASTRON side. *R1, R2*: external researchers. Phases: *(D)* Diagnosing, *(AP)* Action Planning, *(AT)* Action Taking, *(E)* Evaluation, *(SL)* Specifying Learning

In terms of data collection, we created a questionnaire in order to get the experts' views about the applicability of the proposed documentation management approach in domains beyond radio-astronomy. The questionnaire included a section for each proposed feature with the aforementioned introductory material referenced from it, two *likert-scale* questions, and one open-ended one, as follows:

Q1 To what extent could **[this feature]** for ICDs management be transferred to one or more of the application domains/industries you have experienced? **[A six-point likert scale]**

Q2 In your opinion, to what extent **[this feature]** for ICDs management would improve: **[A six-point likert-scale for each related issue to be improved]**

Q3 Please elaborate on your answer to the questions above. **[An open-ended response]**

The selection process of potential candidates for the panel was two-fold. First, we sent invitation letters through email and LinkedIn to the authors of the publicly available ICDs used in the *Diagnosing Phase* of the TAR study (when these were disclosed in said documents). A total of twelve invitations were sent to this group, from which we got four positive responses. Second, we sent invitation letters to the participants of our previous practitioners survey (Cadavid et al. 2020); these participants were already identified as engineers with experience in the architecting of cross-disciplinary Systems-of-Systems. In this case, 50 invitations were sent and we received 8 positive responses; thus we reached an overall total of 12 experts. As illustrated in Table 2 and Fig. 1, the 12 panel participants include staff members of governmental and intergovernmental organizations, research and development centers, and the private sector. Together, they represent significant experience (a mean of 23.4 years) in the domains of *space*, *defense*, *ground transportation* (e.g, railway systems), *energy*, *aeronautics*, *logistics* and *business*. All of them reported experience with ICDs by

Table 2 Current affiliation and years of experience in the industry of the panel participants

ID	Current affiliation	Qualifications	Experience (years)
E1	Software Engineering Institute (USA)	Bachelor	19
E2	DEMCON (NL)	Bachelor	23
E3	European Space Agency - ESA (EU)	PhD	20
E4	Independent consultant (USA)	Master (multiple)	50
E5	Webasto Group (DE)	Master	9
E6	Software Engineering Institute (USA)	PhD	39
E7	Chartered engineer (UK)	Master	30
E8	European Space Agency - ESA (EU)	PhD	15
E9	Software Engineering Institute (USA)	Master	36
E10	NASA (USA)	PhD	15
E11	Warsaw University of Technology (PL)	Master	5
E12	Software Engineering Institute (USA)	Bachelor	20

creating/maintaining them (11 out of 12) or using them as a reference (8 out of 12), or even reverse-engineering them when no ICDs were available (1 out of 12).

4 Results

The findings of both the TAR and the expert panel studies are presented in the following, organized by the research question that they are addressing.

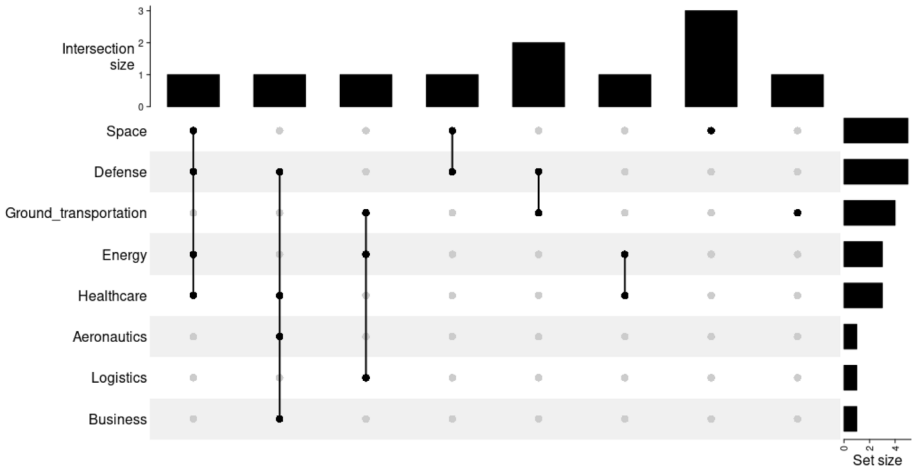

Fig. 1 UpSet diagram of the application-domain expertise as declared by the panel participants. On the horizontal axis, the areas of expertise of each participant. Multiple participants with the same areas of expertise are shown in a single column (where intersection size > 1). On the vertical axis, a representation of each application domain within the expert panel

4.1 RQ1: What are the issues with ICDs management that cause assumptions and misunderstandings when working with these documents in SoS?

The analysis of the results of the virtual focus group, conducted for the Diagnosing phase of the study (as described in Section 3.2), led to a wide variety of interface-management related issues described below. However, although said focus group discussion was geared specifically towards this kind of issues, a number of issues not directly related to interfacing, but to the design decisions of the systems behind the interfaces, emerged from it. These may be out of the scope of the TAR study (and hence will not be considered in the following TAR phases), but they are important for the results discussion; so, they are also described below.

4.1.1 Issues Related to ICD Management

Lack of clarity and cross-domain understandability (CLA) an ICD often contains terminology that may be clear for the people involved in its original version, but years later it could be interpreted differently. This is also the case when multiple disciplines are involved in the process (e.g., hardware and software engineering), as in many cases there are similar terms between these disciplines with a different meaning.
Incompleteness (COM) Some critical details, particularly in the description of the hardware-side of the interfaces, are often omitted in the ICDs. This could lead to risky assumptions, or to error-prone informal information exchange. Additionally, time-behavioral and state-related aspects of the interfaces are rarely included in the ICDs. In particular, scenarios that lead to a failure state are important to include, although it is not always feasible to describe all of them in an ICD.
Lack of uniformity (UNI) The lack of uniformity between ICDs, in the context of an SoS where many interfaces are usually involved, lead to confusion and misinterpretation.
Lack of timely update notifications (UPN) Changes in ICDs are not announced but rather discovered by people while working with them.
Duplicated efforts (NDE) Interfacing-related information is duplicated across ICDs and the artifacts derived from it, e.g., intermediate ad hoc machine-readable formats and other tools created to support the development process. The time and effort required to keep in sync all these information sources is substantial and it could instead be invested in the actual engineering/development activities of the interfaces.

4.1.2 Issues Related to Subsystem Design Decisions

Incomplete designs the physical/logical interfaces sometimes lack elements required by one of the involved parties. A prominent example of this, is the lack of key *monitoring points* (memory locations where specific hardware values can be monitored) on the hardware side, required on the software for proper control of the system state.
Difficult-to-use design some of the interfacing elements required by one of the parties, particularly to access hardware at a low level, are designed in a way that makes their use difficult. For instance, when the hardware design defines multiple *monitoring points* covering the same area. This can be particularly difficult to handle on the software side given the potential contradicting values due to e.g., timing issues when checking these points.
Unbalanced abstraction levels the software (or firmware) layer that provides access to the hardware at a low level (e.g., at a register level) does not always have the right level of abstraction. On the one hand, too little abstraction leads to software tightly coupled with the

hardware, with most of the complexity pushed out to the software side. On the other hand, too high-level abstraction could negatively impact the flexibility that the software can give to the hardware.

4.2 RQ2: What are the Symptoms of the Previously Identified ICD-Management Issues within a Typical Hardware/Software-Oriented ICD?

The following sub-sections describe the particular symptoms identified through the collaborative annotation of a generic ICD template distilled from real hardware/software-oriented ICDs to be used further in the TAR, as described in the Action Planning phase in Section 3.2. These symptoms can be categorized in three types: general ones, specific to a given ICD section, and specific to a given kind of hardware/software interface; the types are elaborated in the following sub-sections.

4.2.1 General Symptoms

The 'general symptoms', identified through the ICD template, correspond to those that do not make reference to a particular section of an ICD, but to the overall document or management process. As a symptom of the lack of timely notifications (UPN), it was pointed out that some people held the naive belief that once an ICD has been written, it is done. Therefore, as ICDs are in fact living documents, *they need to be kept under document control, amended when references change, amended or modified when referred implementations change and so on*. As a symptom of the duplicate efforts (NDE) on ICDs maintenance, it was mentioned that there seems to be a tendency of reiterating information that is written elsewhere. As this is something that is both cumbersome and error-prone, *there is the need for a central knowledge system, (e.g., a lexicon shared throughout the entire organization) that can be queried to supply information that could otherwise be redundant or outdated.*

4.2.2 Symptoms Specific to ICD Sections

The following lists the sections commonly included on ICDs, and the symptoms specific to them, as identified by the participants, and mapped to the issues identified in Section 4.1.1.

Purpose/scope, Audience One-sided description of the interface, which leads to unbalanced views, or to the use of language that is too specific for one of the involved disciplines (CLA).
Referenced documents This section, when not omitted, lacks ready-to-open links to the specific documents referenced (NDE).
Abbreviations This section is often omitted (COM). Abbreviations often overlap with ones from other disciplines, or even change overtime (CLA).
System overview/global description Too much boilerplate content instead of references to relevant program-specific ICDs and documents where deviations from standards are to be expected or where special implementations of protocols are used NDE).
Related requirements Lack of integration with requirements management platforms (COM).

4.2.3 Symptoms Specific to Particular Kinds of Interfaces

The following describe the symptoms identified by the participants along three kinds of hardware/software interfaces that they focused on:

Interfaces between embedded software/firmware and hardware through direct access to peripheral registers

In this kind of interface the software performs operations on the hardware by modifying some of the peripheral's registers values. Likewise, the status can be evaluated by reading other registers values. These registers, consequently, are known as the control and monitoring points of the hardware. The participants emphasized that the ICD user on the software/firmware side needs a register map describing the bitfields (groups of bits with meaning on the hardware side). For this type of interface, participants enumerated the symptoms described in Table 3.

Interfaces between software/firmware and hardware, through physical, low-level data links

In this kind of interfaces, hardware and software components are connected through a datalink bus. In this way, the hardware is operated and its status is evaluated by sending and receiving messages encoded in the 'payload' segments of the data frames defined by the datalink protocol, as illustrated in Fig. 2. Therefore, the ICD user on the software/firmware side needs to know (1) which values should be given for the datalink protocol parameters to send the payload to the hardware end, (2) the possible values sent from the hardware side, and (3) their meaning. For these elements, the internal structure of the payload should be defined using bitfields or textual values depending on the selected encoding. Prominent examples of datalink protocols include asynchronous ones like UART and Ethernet, and synchronous ones like I2C, SPI, and CAN. Table 4 lists the symptoms pointed out by the participants as common when working with this kind of hardware/software specifications.

Interfaces between hardware and software, through higher-level Network and Transport protocols

In this kind of interfaces, hardware is accessible through protocols defined on top of layers of higher-level network and transport control, such as TCP/IP or UDP/IP. As these base protocols require less custom configuration, and most platforms already support them, the information provided to the ICD user on the software side is limited to the payload structure, as shown in Fig. 3. Examples of this kind of interface include protocols that work on top of TCP/IP like OPC-UA, MQTT, CoAP. Table 5 lists the symptoms as identified by the participants.

Table 3 Particular symptoms of the ICD-management related issues, identified by ASTRON engineers, for interfaces between software/firmware and hardware through registers

Issue(s)	Symptom
COM, UNI	Missing hardware-related details: • values (registers' default value) • Operations atomicity • Need for bit alignment • Atomicity of byte writings • Mandatory timeouts for hardware operations • Registers' signedness • Filler bits/bytes purpose
NDE	Register offsets scattered cross ICDs and source code

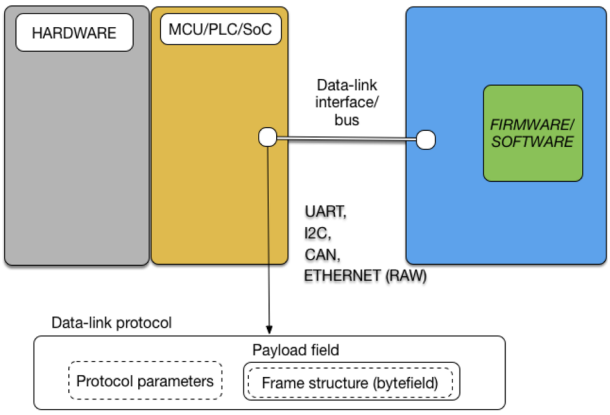

Fig. 2 Typical hardware/software interface through a data-link bus. Input and output operations on the hardware are performed by sending commands and parameters (which follow a custom, commonly agreed format and semantics) through a physical hardware communication protocol

4.3 RQ3: What are the Features Rrequired for a DaC-Based ICD Management Approach to Address such Issues?

In the Action Planning phase, four features were defined addressing the issues identified in the previous phase:
Documentation-oriented quality gates QC enforcing certain minimum quality criteria for the ICDs through centrally-managed rules. This involves automating the publication process of the ICDs through a CI/CD platform, with publication rules (i.e., quality gates) tailored to the context of technical documentation.
Embedded machine-readable-formalisms F integrating existing formalisms for the description of technical elements within ICDs, particularly hardware-oriented ones. Such formalisms, when embedded in an ICD, can be automatically transformed into human-readable content (e.g., sections within the generated documents) and other artifacts (e.g., libraries).

Table 4 Particular symptoms of the ICD-management related issues, identified by ASTRON engineers, for interfaces based on low-level data links

Issue	Symptom
COM	Missing protocol/encoding related details: • Endianness • Filler bits/bytes specification • Signedness • Presence or absence of acknowledge (ACK) messages • Time to completion
CLA	Text that contradicts information on tables (single source of truth)
NDE	Command descriptions that require a manual (error prone) mapping of each parameter index into a certain byte/bit position when working on the software side. Error-prone re-mapping when a parameter update takes place.

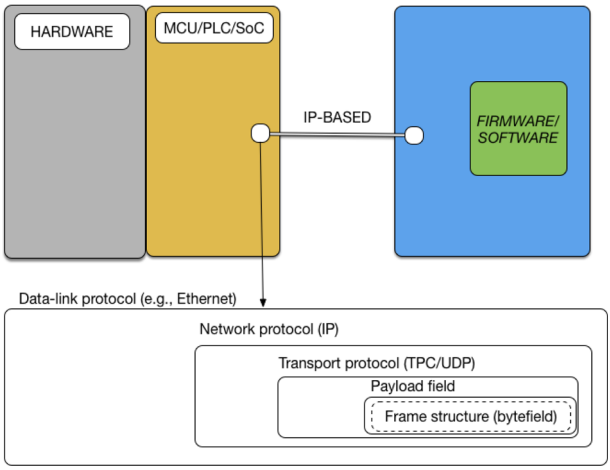

Fig. 3 Hardware/software interface through a higher-level communication protocol. In this interface, input and output operations are performed by encoding commands and parameters on an existing communications protocol (in the example, TCP/UDP)

Document macros for automatic content generation M providing custom extensions for the selected lightweight markup language used for ICD definition allowing engineers and technical writers to define where and how content will be automatically generated within an ICD.

Centralized-documents tracking DT keeping track of the publication status of the ICDs within the project/organization and of the dependencies between them. As this feature cannot be fully implemented through the version-control system (e.g., Git) itself, it entails creating a custom platform that would work in tandem with/on top of the CI/CD platform.

Table 5 Particular symptoms of the ICD-management related issues, identified by ASTRON engineers, for interfaces based on high-level network and transport protocols

Issue	Symptom
COM	Missing protocol/encoding related details: • Is timing relevant • Signedness • Endianness • Lack of communication examples • Timeouts on the hardware side • Undocumented incompatibility with higher-level protocol
NDE	• Like in the monitoring and control registers described in the previous section, network and transport protocols also involve the mapping of byte/bit positions, but in this case, within the protocol payload. Consequently, re-mapping when a parameter update takes place is an error-prone process. • Restating information within the ICDs that is available somewhere else (and that should be referenced instead).

Table 6 describes particular symptoms of the issues identified through RQ1 and RQ2 and how the proposed features plan to address them.

4.4 RQ4: What is the Design of an ICD Management Pipeline that Provides the Identified Features?

In the Action Taking phase, a solution was implemented as a functional proof of concept of a DaC pipeline instantiating the features identified in the previous phase. The pipeline is depicted in Fig. 4, and is described as follows. First, the writing/editing process of an ICD source document (1) is supported by any text editor since a text-based lightweight markup language is being used. For the proof of concept, Asciidoc was chosen as the lightweight markup language considering (in addition to its extensibility through custom macros) the variety of environments it can be interpreted on[9] and the flexibility this could provide to the solution on this, or follow-up cycles of the study. The collaboration between writers is consequently mediated through a version management system like Git. Once a new ICD is ready for publication, the authors tag it as an official version. When such a tag is set, a CI/CD platform, linked to the ICD repository (2), launches the publication process using a centrally managed (at the project/organization level) building environment (3) that validates the minimum quality criteria in the process through **Documentation-oriented quality gates** (QC). This building environment, in turn, includes a number of custom extensions to enable the ICD-related **Document macros** (M) and in particular the ones that deal with the **Embedded machine-readable formalisms** (F). The events generated by the previous steps are reported to an API (4) as the means to have **Centralized document tracking** (DT) of the overall documentation status, and to provide context information to the building process of other documents (e.g., when dependencies between documents are involved).

Finally, the documentation and other generated artifacts are kept as the single source of truth for each interface by storing them in a canonical location (i.e., a URL accepted as the official one). With these locations the documentation users (e.g., software developers), besides accessing the latest version of an ICD, can integrate automated checks in their development environment (5) to verify that software artifacts are up-to-date with respect to the documentation they were generated from. The documentation management platform (6), on the other hand, has two purposes in the above process. On the one hand, for engineers or technical writers to have access to easy-to-read information about errors or failed quality gates during the ICD publication process. On the other hand, to serve as a dashboard that provides an overview of the overall documentation status.

In the following, the elements of the ICD-management pipeline that needed custom tools, tailored to suit the features identified in the previous section (RQ2), are described.

4.4.1 Quality-Gates and Markup Language Extensions

There are many tools that can perform a wide variety of quality checks in software source code within a CI/CD platform. However, when it comes to the source code of documentation, that is, markup language documents, quality criteria like the one defined in the Action Planning phase (RQ2) require extending the interpreter of such a markup language so that the criteria are evaluated while the document is being processed. These extensions (e.g., the *custom macros* (M)) need not only to properly report their status during the building process (e.g.,

[9] For example, in front-end or backend, using https://github.com/asciidoctor/asciidoctor.js*Asciidoctor.js* and https://github.com/asciidoctor/asciidoctorj*AsciidoctorJ* respectively.

Table 6 ICD-management issues, their observed symptoms, and the identified associated features

Issue	Symptom	DaC feature
CLA	Overlapping/colliding abbreviations and terms.	(M) macros to (1) insert references to a centralized glossary, and (2) to generate a glossary section within the document accordingly. (QG) for undefined abbreviations
	Unclear language.	(QG) number of violations to the writing style rules defined by the organization
	Broken links.	(QC) Broken links metric.
COM UNI	Missing details: on specifications for reading/writing from/to a peripheral (e.g., endianness, R/W rights, update rates, etc.).; on atomicity (e.g., which operations are atomic); on timing (e.g., commands timeout, ack signals timeout, etc.)	(F)(M) Support for hardware-oriented formalisms and (QG) to validate the completeness of instances of such formalisms
UPN	ICDs should be amended when referenced documents are updated.	(DT) Identify and notify when a document has a more recent timestamp than the ones that referenced it
	Changes are discovered rather than announced; outdated documentation.	(F) Support for hardware-oriented formalisms and (M) macros for the generation of software artifacts (e.g., headers) and human-readable sections from the machine-readable formalisms. Based on this, and the (DT) platform, the artifacts' checksum (e.g., headers) previously generated by the documentation pipeline, and currently used in the development environment, could be automatically compared, during the (software) building process, against the most recent one (available online)
NDE	Error-prone process of building a command's payload, (e.g., mapping parameter bits within a given index).	(F) Support for hardware-oriented formalisms and (M) macros for the automatic generation of software artifacts (e.g., library headers) and human-readable sections from the machine-readable formalisms
	Information replicated over multiple ICDs (no need to reiterate what is written elsewhere).	(DT) tracking of the multiple versions of an ICD (sources and published builds) and (M) for referencing versions of other ICDs, reliably, through the centralized document tracking platform, and generating references section automatically
	Need for the right balance between documentation maintenance efforts and actual engineering/development ones.	(F) Support for hardware-oriented formalisms and (M)macros for the generation of software artifacts (e.g., headers) and human-readable sections from the machine-readable formalisms.

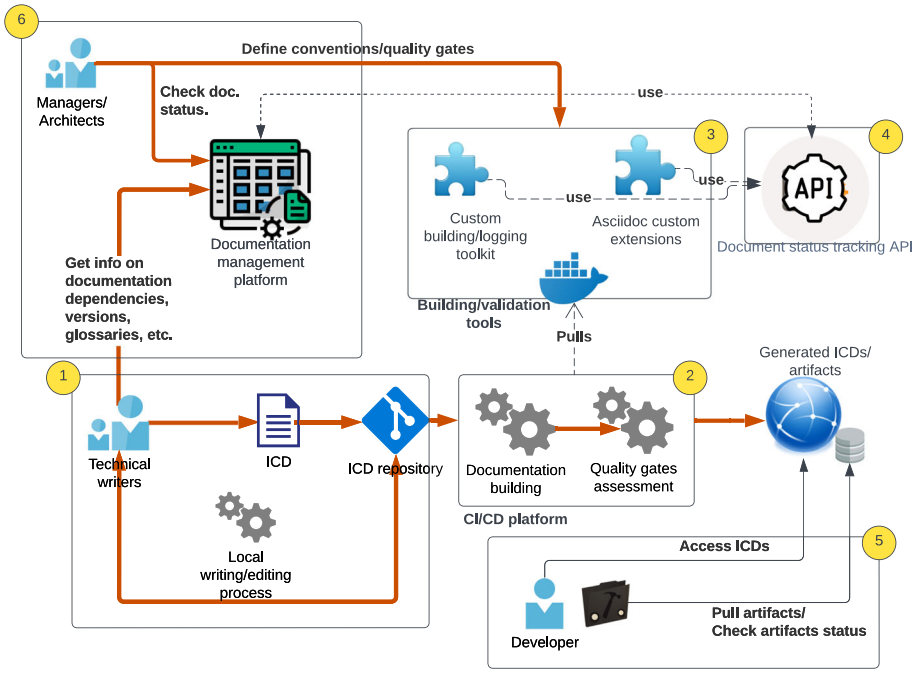

Fig. 4 General overview of the ICD management pipeline

distinguishing between failed quality criteria and syntax errors) but also to work in tandem with the API ((4) in Fig. 4). For this reason, a toolkit for building extensions with such features was created on top of the AsciidoctorJ[10] platform, and used to build the artifacts depicted in Fig. 4 (3). More details are reported in the project repository[11].

4.4.2 Centralized Document Tracking

For the proposed DaC-based ICD pipeline, the generated documents are expected to become the single source of truth for all the development and testing tasks conducted on the interfaces described by it. Therefore, keeping track of the official documents and the sources from which they were generated is key to their proper evolution over time. On top of that, an ICD is often dependent on a specific version of other ICDs it is based on, and hence the validity or consistency of the former could be affected by changes in the latter. Given this, a documentation-status tracking API was implemented for the proposed documentation pipeline. This API, working in tandem with the aforementioned building components, keeps track of the locations of the documents and artifacts generated from each ICD, their dependencies, and their corresponding sources. Furthermore, it updates the status of the documents given the events captured by it as illustrated in Fig. 5, which, in turn, can be accessed through its front-end application ((6) in Fig. 4). More details are reported in the project repository[12].

[10] https://docs.asciidoctor.org/asciidoctorj/latest/

[11] https://github.com/hcadavid/asciidoc-icd-extensions

[12] https://github.com/hcadavid/documentation-dashboard

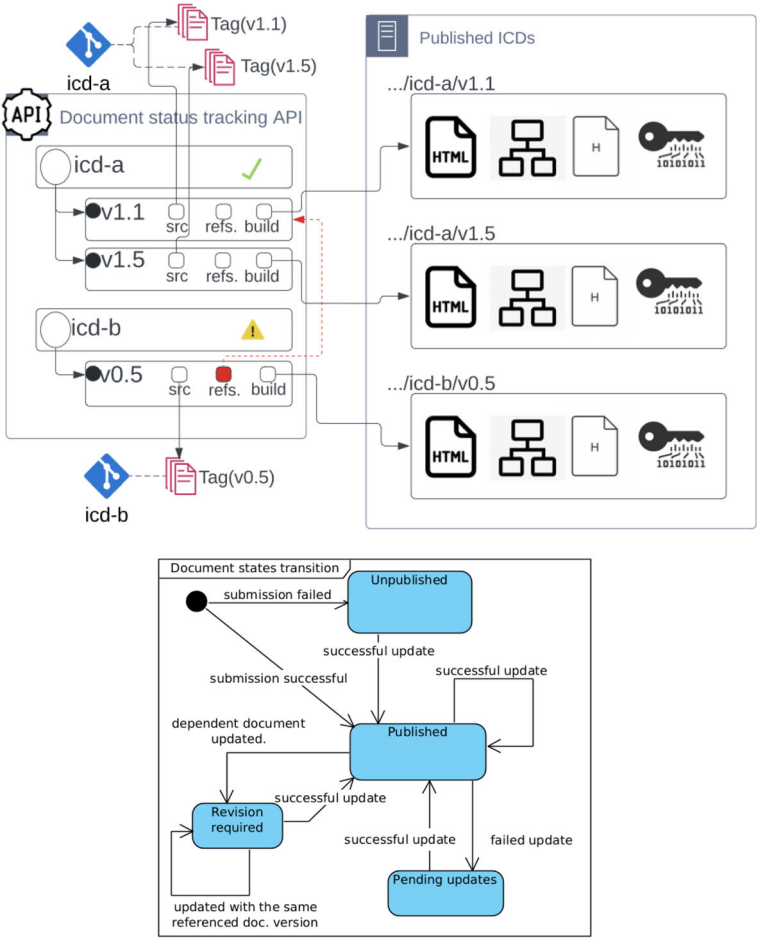

Fig. 5 The diagram of the bottom depicts the status transitions of the ICDs indexed in the centralized document tracking API. On the top left, the information indexed for each ICD is shown: (src) sources code, (refs) references to other ICDs, (build) canonical location of each version's generated document. Each published ICD (on the top right), contains the generated documents, artifacts, and their corresponding unique checksum. In this particular case, document 'icd-b' has a status of *REVISION-REQUIRED*, as it is based on version 1.1 of 'icd-a', which is no longer the most recent one

4.4.3 Macros for Content Generation & Embedded Machine-Readable Formalism

SystemRDL[13], a widely adopted hardware description language was chosen to explore the use of a machine-readable formalism embedded in ICDs to describe hardware elements of interfaces (F). This involved the creation of a SystemRDL macro for Asciidoc that performs validations according to the defined completeness quality-gates, and generates a human-readable representation of the hardware along with base software artifacts (as depicted in Fig. 6). With the automatic generation of check-sums of the said generated artifacts imple-

[13] https://www.accellera.org/activities/working-groups/systemrdl

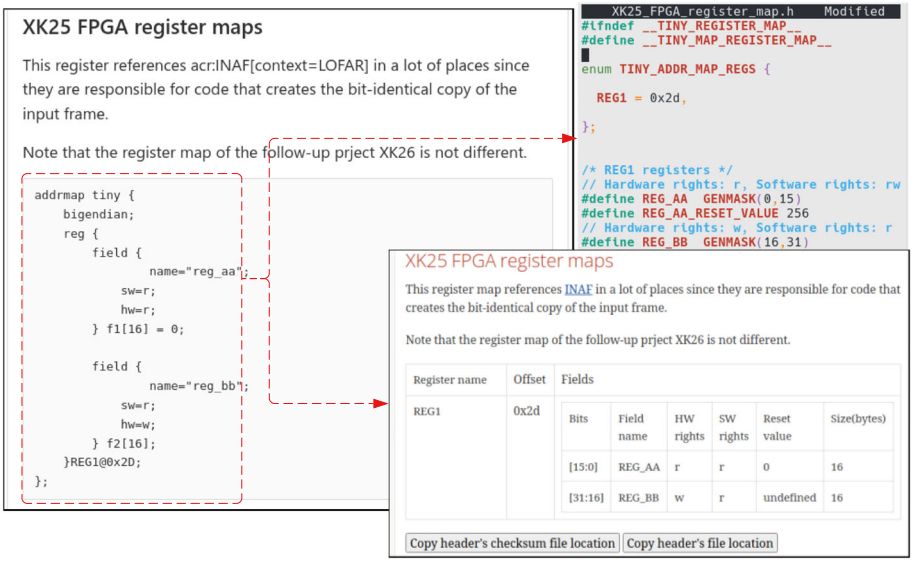

Fig. 6 Excerpt from the source of a document created during an evaluation exercise (left), the generated source code headers (top-right), and the human-readable section expanded in the generated document (bottom-right)

mented in the extension, developers can check whether their local copy is up to date, as previously described.

Additional content generation extensions, based on the results of the Action Planning phase of the study, include: automatic generation/validation of the document log section (based on Git history), references, and glossary. The source of a sample ICD written in Asciidoc, using the above extensions, and its corresponding output, is included in the replication package[14].

4.5 RQ5: To what Extent can the Designed ICD Management Pipeline Improve the Identified ICD Management-Related Issues?

This research question deals with the efficacy and fitness for purpose of the features identified in the Action Planning phase and implemented in the Action Taking phase. As the evaluation instrument used for answering this research question was derived from the results of RQ1, this Section elaborates on the said instrument details (Section 4.5.1), before introducing the analysis of the results obtained from it per feature (Sections 4.5.2, 4.5.3, 4.5.4 and 4.5.5.)

4.5.1 Evaluation Instrument

The purpose of the single-case mechanism experiment (as described in Section 3.2.1) was to assess to what extent, according to the ASTRON engineers' viewpoint, the proposed features (RQ3), and the way they were implemented (RQ4) would deal with the ICD-management related issues (RQ1) and their particular symptoms (RQ2). To this end, the functional proof-of-concept of the implemented features was configured to work with an infrastructure similar to the one used by ASTRON engineers: GitLab for version control system, GitLab CI/CD

[14] https://figshare.com/s/671cc7450d8a6b8e2987

for creating documentation pipelines, and GitLab pages for publishing the generated ICDs. An instance of the platform that keeps track of the published ICDs was deployed and made available on the Internet, while the building/validation tools and extensions were published as a Docker container so these could used by the documentation pipeline anywhere. In order for the participants not only to understand but also to experience the four proposed features, they were asked to carry three hands-on tasks assuming different roles, as they would do in a real setting.

In the first scenario, participants were introduced, hands-on, to the required configuration activities, the editing with a lightweight markup language, and the basics of ICDs versioning. In particular, they were asked to setup a documentation pipeline for a new ICD in the CI/CD platform, and to set up and generate two sequential versions of it: the 'staging' version (to be shared and reviewed) and the 'canonical' one, which would be indexed by the platform as such.

In the second scenario, the participants were asked to include on the previously registered ICD: (1) the specification of a low-level hardware interface (i.e., a register map) which includes a memory address inconsistency; and (2) a paragraph that violates a hypothetical writing style rule. While dealing with the errors reported by the pipeline (unfulfilled 'quality gates') using the details provided by them, the participants experienced: (1) what kind of quality criteria can be enforced for both formal and text-based elements; (2) how these embedded formal elements could reduce redundant and error-prone writing/transcription efforts through content/code generation.

In the third scenario, participants were asked first to: (1) create new ICDs and add them as references (dependencies) to the one created in the previous scenarios; and (2) use the software headers generation feature from such ICDs, and by assuming a developer role to create a basic development environment with it. Afterwards, they were asked to make changes on the hardware specification within the ICD (created in scenario two), and on the new ICDs referenced by it. Then they were asked to deal with the warnings generated by the platform (about the need for revisiting an ICD with outdated dependencies) and in the development environment (about the use of headers generated from an outdated specification). This scenario is aimed at illustrating how the ICDs, by becoming the actual 'single source of truth' through the centralized index, could be used to foresee potential issues as interfaces evolve over time. In particular, how keeping track of the documents' life-cycle and dependencies between them, would enable the timely notification of changes.

Once the practitioners completed the exercise[15], they took an online survey where they evaluated the efficacy and fitness for purpose of the proposed features. The analysis of said evaluations is described in the following sub-sections.

4.5.2 Documentation-Oriented Quality Gates

The practitioners' perception on enforcing minimum quality criteria by means of quality gates was in general positive, as shown by questions Q1 and Q2 in Table 7. However, the *Disagreement* and *Neutral* responses in Q1, about the enforcement of the style of the ICD elements described in natural language, were justified with doubts about its applicability in a more realistic setting. More specifically, two of the participants pointed out that (1) some text-based descriptions could be very hard to validate, if such validation is at a technical level, and (2) that writing styles are hard to impose and writers could find workarounds to avoid them. The enforcement of technical elements of the ICDs like forgotten keywords and

[15] Available at https://github.com/search-rug/doc-as-code-single-case-experiment

Table 7 Perceived applicability of the DaC features after using them in the proof-of-concept

	Question	Participants P1	P2	P3	P4	P5
Q1 (QG)	I believe that enforcing the expected minimum quality criteria of the texts within the ICDs, by means of automated 'quality gates', would improve their clarity and understandability.	A	D	SA	SA	N
Q2 (QG)	I believe that enforcing the expected minimum quality criteria of the technical details included on the ICDs (e.g., hardware-level details), by means of automated 'quality gates', would reduce the documentation-related integration/operational issues experienced in the past.	A	A	SA	SA	N
Q3 (M)	I think that having part of the documentation (e.g., glossaries, hardware description, etc) and other artifacts (e.g., software components) generated from models embedded in the ICDs would reduce the documentation maintenance efforts when compared with the existing approach.	A	N	SA	SA	SA
Q4 (F)	I think the automatic generation of human-readable content and software artifacts from formal models embedded in the ICDs (e.g., SystemRDL, ARGS, etc.), would improve the data-transcription-related issues experienced in the past.	A	A	A	SA	SA
Q5 (DT)	I think that having centrally managed ICDs, with their different versions/dependencies tracked, and status changes reported to relevant roles, would be helpful to mitigate the issues related to outdated documentation experienced in the past.	A	A	A	SA	SA
Q6 (DT)	I think that integrating the information provided by the centrally-managed ICDs into the software development process, in a realistic setting, would be useful to avoid shipping software based on ICDs that could be outdated or that need to be revisited.	A	N	A	SA	A

Grades: (SA) Strongly Agree, (A) Agree, (N) Neigher agree nor disagree, (D) Disagree, (SD) Strongly disagree

wrong ranges, on the other hand, was seen in general as useful, although some interfacing parameters could be too difficult to validate automatically, especially the ones related to dynamic behavior, time dependencies and performance limitations. Overall, with the ICDs seen as the single source of truth and reference for engineers to work on an interface, the need for these automated quality gates would decrease in the long run, as will the time spent on resolving inconsistencies during development and testing.

4.5.3 Document Macros for Automatic Content Generation

According to the justifications for the grades given in Q3, having macros for content generation would help ICDs to become the single source of truth that can be referenced throughout the process. This would help, in turn, in identifying incorrect assumptions in the early stages of the process. On the other hand, the elements automatically generated through macros tailored for ICDs (content, artifacts, etc.), as long as they are reliable and reproducible, would not only improve their consistency but also free time to work on the content that cannot be automatically generated. However, although the above would improve the overall quality of the documentation, that would not necessarily be the case for its maintainability. In this regard,

the analysis of Slack conversations during the evaluation exercise highlighted two areas of improvement. First, as a means to improve the workflow speed, an editing tool tailored to the proposed features, that is to say, that allows technical writers to identify errors or unfulfilled quality criteria before submitting changes to the documentation pipeline was deemed necessary. Second, the pipeline should support allowing locally-defined, document-specific glossary entries, in addition to the ones defined at project/organization level.

4.5.4 Embedded Machine-Readable Formalisms

Question Q4 (see Table 7) and the corresponding open-ended responses also showed an overall positive perception on the applicability of an embedded formalism within the ICDs as a means to remove ambiguity through the uniformity and standardization of the artifacts automatically derived from it. This is seen as a must-have to prevent issues in different engineering groups. Furthermore, this feature was perceived as something that would save a considerable amount of time normally spent writing boilerplate ICD parts, and reduce the human factor that often causes issues when translating from ICD to source code.

4.5.5 Centralized-Documents Tracking

According to the practitioners' views on Q5 and Q6, a centralized platform makes sense if ICDs are expected to be the single source of truth within a project/organization. Moreover, adopting this systematic approach for managing ICDs implies that now it would be necessary to decide, at a higher level within the project/organization, when to go from one version to another across the whole product. Although this would add complexity to the overall process, it would be a positive change. On the other hand, the generation of artifacts with version matching as supported by such a platform would prevent developers from using an outdated version when implementing parts of an interface. However, as pointed out by the participants, to properly close the loop failed version test results should be reported to the central ICD hub so that action can be taken. Furthermore, the tooling should also allow the interfaces to be round-trip converted from source-code back to the model. On top of these observations, practitioners emphasized that although the documentation pipeline tooling itself would prevent small but annoying mistakes, most of the success of the interfacing process would still lie on the communication and cooperation between engineers. Moreover, it would also be important to ensure that engineering and maintenance teams also keep this documentation up-to-date during operations.

4.6 RQ6: To what Extent can the Proposed ICD Management Approach be Transferred from the Radio-Astronomy Domain to other Application Domains?

This section elaborates on the evaluation performed through the expert panel described in Section 3.2 and is presented in terms of the proposed features: *Embedded machine-readable formalisms* (F), *Document macros for automatic content generation* (M), *Documentation-oriented quality gates* (QG), and *Centralized-documents tracking* (DT). As the feature of *automated content generation* depends on *embedded formalisms* (something that can be specified with these formalisms, can also be automatically generated), they are discussed together.

4.6.1 Embedded Machine-Readable Formalisms and Document Macros for Automatic Content Generation

The features of *embedded formalisms* and *automated content generation* for ICDs management, were deemed as transferable, albeit to a different extent, ranging from 'a little' to 'a great extent' (Question 2.1). As seen in the top of Fig. 7, in the applicability scale the opinions are equally divided between 'a lot'/'to a great extent' and 'a little'/'to a moderate amount'. This is for the most part also the case when looking at each application domain individually, as seen at the bottom part of Fig. 7.

As described on Fig. 8, experts ranked uniformity as the aspect that would be improved the most, out of these features, followed by completeness (Question 2.2). The foreseen improvements on duplicate efforts were also ranked high, although three respondents selected the 'do not know' option as according to the responses to Question 2.3 they had no elements to assess it.

By analyzing the open-ended responses that experts gave to Question 2.3, three themes emerged: (1) arguments for why it would make sense to adopt the two proposed features in the domains they have experience on, (2) what should be considered before adopting these features, regardless of the application domain, and (3) potential limitations upon their adoption. These three themes are elaborated in the following paragraphs.

First, experts pointed out that the one of the premises of the proposed approach, namely the fact that interface descriptions tend to be defective is correct (E4). These defects seem to be prominent when dealing with system-level interfacing elements, as any change always has cascading effects on related subsystems (E1). Specific examples of these cascading effects were given, such as the ICDs with UDP[16] specifications (E1): when the bitfield defined for the payload is modified, it requires updating *offset* values all over the place (source code, documents, etc.). Moreover, experts already had the perception that versioned databases of interface definitions would improve completeness, reduce human errors, and ease the regeneration of documents upon updates (E3) as well as the system verification (E9).

Second, other experts indicated that the applicability of the features would be subject to a number of factors. These factors include the availability of DSLs for the particular elements considered in the interface, the expertise of the ICD designer on them (E6), and an agreement between the stakeholders involved in the system on adopting this approach (E10). For the latter, it was highlighted that having different documentation approaches, or using them with different levels of effort would lead to further issues in the architecting process. Furthermore, in the end, the applicability of *embedded formalism/content generation* would also depend on the level of formalism required by the project (E10).

Finally, from the comments focused on the limitations of these two features, the experts pointed out that the scenario used in the evaluation is focused on what is transferred through the interface, but not how it is done (E5), e.g., re-negotiating hand-shakes when a message exchange fails. That is to say, the lack of dynamic behavior specifications becomes an issue here. On the other hand, in many cases, particularly in the avionics and automotive industry, it is common to maintain interface information in databases. Therefore, rather than the proposed embedded text-based formalisms, this approach would be applicable only by enabling references to said databases within the ICDs (E5). It is also worth noting that a number of experts (E8, E11, E12), indicated that these features would *improve* correctness (by providing formal elements, complementary to the natural language), uniformity, and completeness, but they may not *guarantee* the latter by default. Human intervention would be needed,

[16] User Datagram Protocol

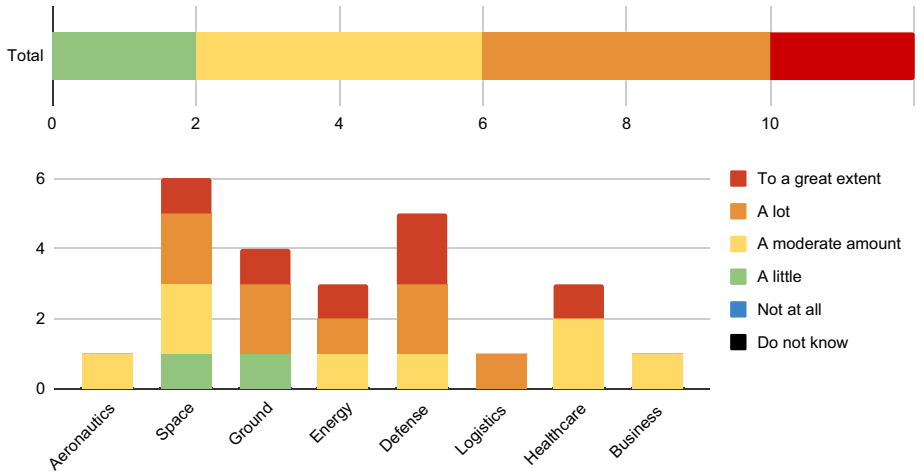

Fig. 7 Distribution of the responses of Q2.1: *To what extent could the features of embedded formalisms and automated content generation for ICDs management be transferred to one or more of the application domains/industries you have experienced?*. Top: responses per expert. Bottom: responses distributed across experts' application domains

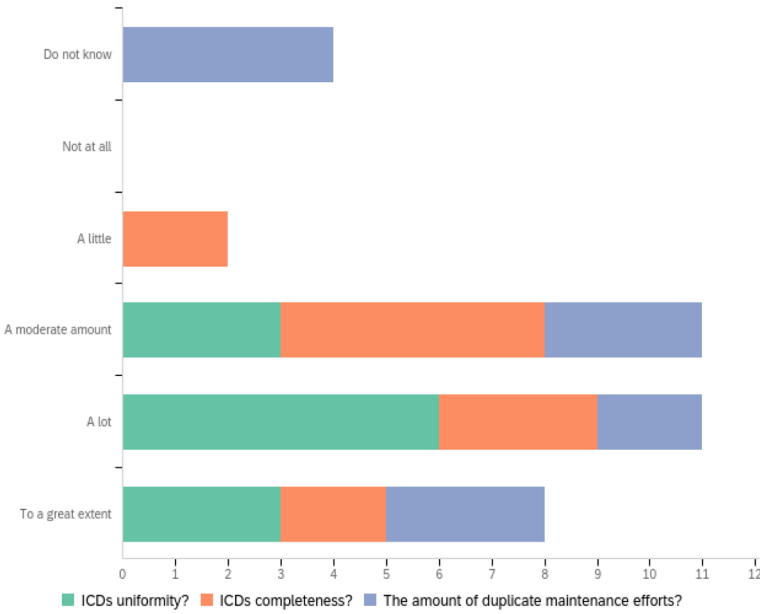

Fig. 8 Distribution of the responses of Q2.2: *In your opinion, to what extent the features of embedded formalisms and automated content generation for ICDs management would improve ICDs uniformity, ICDs completeness, and reduce the need for Duplicate efforts?*

for instance, by distilling guidelines or checklists (and feeding them to the platform) from previous experiences creating similar interfaces in the given context/domain.

4.6.2 Documentation-Oriented Quality Gates

The feature of *enforced quality criteria through quality gates* was also seen as transferable to other domains by all the experts (Question 3.1), but to a larger extent (per participant and per area), as seen on Fig. 9. In this case, given the responses to Question 3.2, incomplete specifications is seemingly the problem that this feature would improve the most, as shown in Fig. 10.

The analysis to the open-ended question related to this feature (Question 3.3) also leads to the three themes identified for the previous two features, namely the arguments about why it makes sense to adopt the features, considerations for its applicability, and potential limitations. However, consistently with the results of Question 3.1, the first two themes (which are positive in general) are the most prominent ones. We elaborate on these three themes in the following paragraphs.

First, the experts highlighted the need for quality rules defined up-front for this kind of electronic documents (E4, E5), particularly given the variability on their quality (E2) — across engineers and disciplines — and level of detail (E3). This would be particularly important at the last stage of the ICDs formal update processes (E12). Furthermore, bringing these 'quality gates' from the software domain to the other ones, e.g., systems engineering, *could help all the disciplines involved to speak the same language* (E10). It is also worth noting that a number of experts expressed the same opinion on *the importance of the quality gates and artifacts included for ensuring proper acronyms definition and allowing the automatic generation of glossaries*. In particular, they mentioned that this terminology-related inconsistencies are common (E3) specially when there is no central authority that manages them (E6); hence, just quality-checking them would improve many of the ICDs they have worked on (E10).

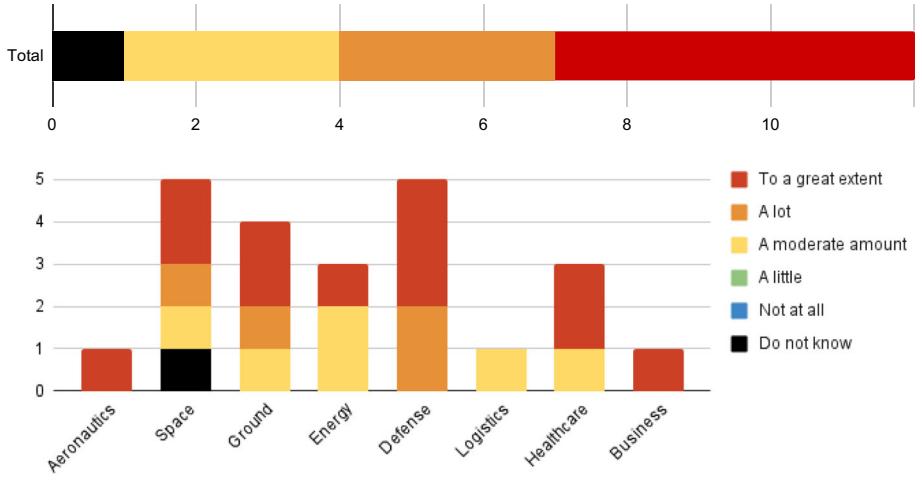

Fig. 9 Distribution of the responses of Q3.1: *To what extent could the enforcement of the expected quality criteria of ICDs by means of Go/No go Quality gates be transferred to one or more of the application domains/industries you have experienced?*. Top: responses per expert. Bottom: responses distributed across experts' application domains

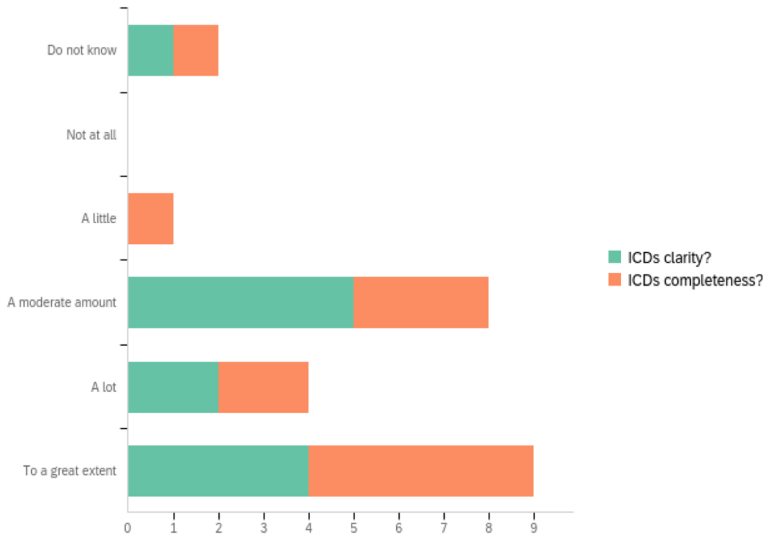

Fig. 10 Distribution of the responses of Q3.2: *In your opinion, to what extent enforcing the expected quality criteria of ICDs by means of Go/No go Quality gates would improve their clarity and completeness?*

Second, regarding the considerations for the applicability of these documentation-oriented quality gates, interface requirements were mentioned as an ambiguity-prone, commonly used, ICD element in multiple domains. Therefore, the quality of said requirements should also be enforced through quality gates using, for instance, existing rule-based NLP tools that assess their quality (E4). Furthermore, in concordance with one of the comments about the *embedded models* and *content generation* features, the quality gates should also be able to check naming consistency within the documents against the parameter-databases commonly used in avionics and automotive industries (E5).

Third, as potential limitations, the experts indicated that adopting this automated validation process in any domain could lead to wrong assumptions about the reliability of the document, which would not be necessarily guaranteed by the enabled gates (E9). To avoid this, *manual quality processes should still be in place as part of the documentation life cycle*. Finally, one of the experts (E11) was of the opinion that selecting the criteria for the quality gates, could be impossible in complex ICDs, particularly when clarity (beyond terminology consistency) is what needs to be enforced.

4.6.3 Centralized-Documents Tracking

According to the experts' viewpoints on Question 4.1, the feature of *Centralized-documents tracking* is, in comparison to the previous ones, the more transferable one, as seen in Fig. 11 (with 10 out of 12 scores betwen 'a lot' and 'to a great extent'). According to Fig. 12, however, the experts were almost evenly divided between its applicability as the *single source of truth* and as a means to reduce *duplicate maintenance efforts*. Nevertheless, from the analysis of Question 4.3 (*Please elaborate on your answer to the questions above*) a prominent theme emerged from the comments of seven out of the 12 experts that would explain most of the aforementioned positive views on transferability: the dependency-tracking capabilities this feature enables between documents. In particular, two of them (E5, E8) mentioned that said

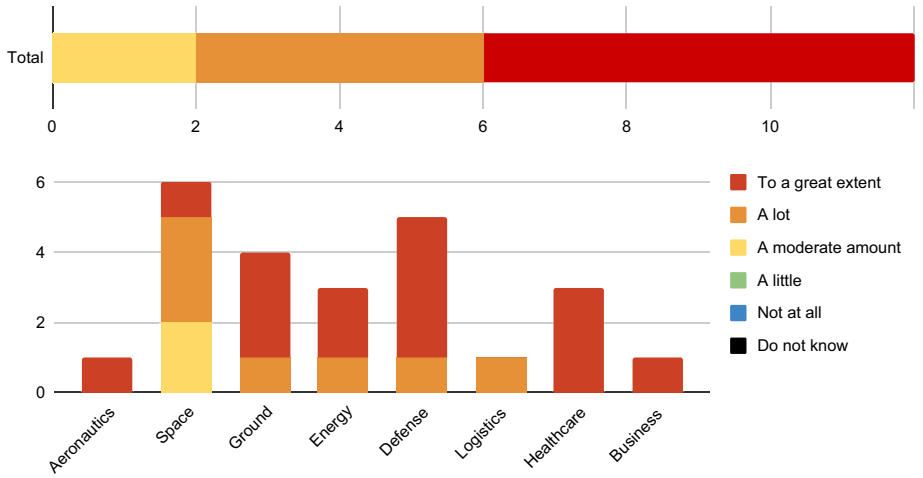

Fig. 11 Distribution of the responses of Q4.1: *To what extent could the centralized ICDs index/tracking feature (including the ability to make references between documents) be transferred to one or more of the application domains/industries you have experienced?*

capabilities are promising for ICDs as similar capabilities have shown their value in the context of requirements management tools like DOORS[17]. The other five pointed out that *keeping track of specific definitions within the ICDs and not just the whole ICD, and in addition to the quality check (E6), would be of great value to improve traceability and to ensure proper actions are taken when a change is made*. These definitions include interface requirements, diagrams and data schemas (E4, E10, E12). Furthermore, *tracking the changes on these definitions by the subsystem/capability they belong to would be key in keeping said systems up to date* (E1).

Two remaining (and less prominent) themes were identified from the experts' viewpoints on Question 4.3, namely applicability benefits and applicability considerations. The former corresponds to three viewpoints that highlight the centralized index feature as useful to make sure that changes will not go by unnoticed (E9, E11), particularly in cases where there are different versions of a platform for which the interfaces are defined (E2). The latter includes remarks from two experts: first, in some cases *the process of updating an ICD is very formal, and hence there is no actual risk of having multiple versions of ICDs* (E12). Second, a centrally managed ICD cannot be, strictly speaking, considered as a single source of truth given the numerous (and independently developed) models or views it puts together, but rather an 'authoritative' source of truth.

5 Discussion

In this work first a technical action research study was conducted with a group of ASTRON engineers in the context of the LOFAR project, aimed at designing and evaluating a potential treatment for the ICD-management issues identified in previous studies(Cadavid et al. 2020; Cadavid 2021). During the *diagnosing* phases of the TAR we identified *uniformity*, *timely*

[17] IBM Engineering Requirements Management tool https://www.ibm.com/docs/en/ermd/9.7.0?topic=overview-doors

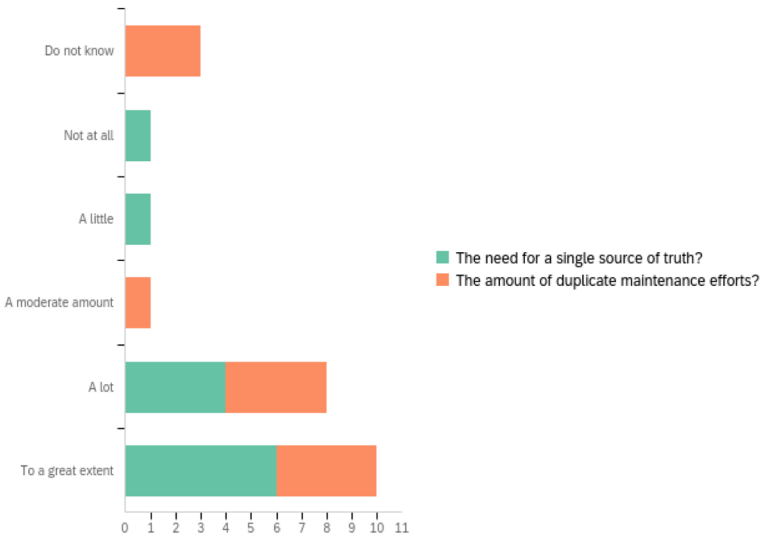

Fig. 12 Distribution of the responses of Q4.2: *In your opinion, to what extent the centralized ICDs index/tracking feature would improve the need for a single source of truth and the amount of duplicate maintenance efforts?*

update notifications, and *duplicate maintenance efforts*, as important issues to be addressed in addition to the ones unveiled by said studies **(RQ1)**, namely the lack of *completeness* and *clarity*. Furthermore, in the *action planning* phase, we gathered insights on the particular symptoms that these issues are commonly exhibiting on ICDs **(RQ2)**. These symptoms guided the identification of four features aimed at improving the aforementioned issues **(RQ3)**, which were then translated into a ICD management platform design and a functional proof-of-concept **(RQ4)**.

The efficacy and fitness for purpose evaluation of these features by ASTRON participants **(RQ5)** suggest that, by enabling an ICD to be the actual single source of truth, they can indeed prevent wrong assumptions early in the ICD creation process, ensure its uniformity, and improve its overall quality. Furthermore, these features are seemingly capable of addressing similar issues in domains beyond ASTRON's radio astronomy one —in systems whose engineering processes are already partially or fully document-centered— as suggested from the insights collected from a panel of experts **(RQ6)**. In particular, according to the areas of expertise of this panel members, this seems to be the case across domains where actual SoS are created (as identified in Cadavid et al. (2020)), such as *ground transportation*, *robotics*, *aerospace* and *aeronautics*.

The following sub-sections elaborate on the implications of these results, for both practitioners and researchers.

5.1 Human Factors in Interface Management

Human factors were a prominent element across the TAR and expert panel evaluations (summarized in Fig. 13), as they were seen as key to ensure reliable interface specifications, regardless of the approach used for managing them. One of these factors was the design

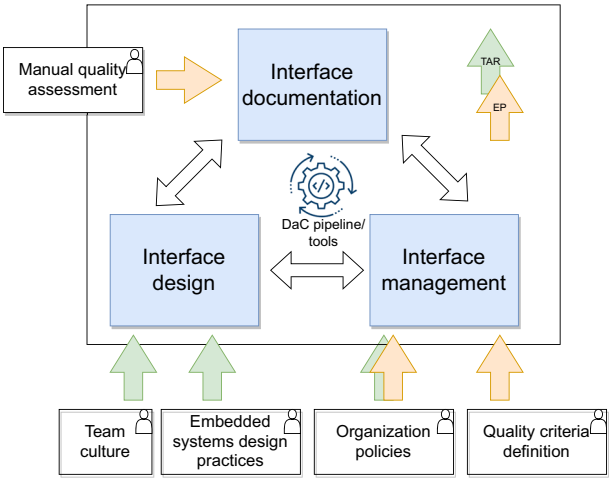

Fig. 13 Context in which the documentation management approach takes place, and the human factors to be considered for it to have the expected impact on the identified issues. The arrow colors indicate from which part of the study the given factors where identified: the technical action research (TAR), or the expert panel (EP)

practices followed by engineers working on the hardware side, as they are often inconsistent or difficult to understand. Therefore, identifying and encouraging **best design practices**, e.g., having 'balanced' abstraction levels as discussed in Section 4.1.2, could prevent potential misinterpretation of interface specifications. Likewise, promoting **team culture**, by encouraging collaboration and communication between the involved parties (as already highlighted by previous studies (Cadavid et al. 2020; Fairley 2019) would improve the quality of the designs and the clarity of the documentation of their related interfaces, by providing a proper two-sided description of them. In this sense, this ICD management approach *would provide a solid foundation for such a collaboration, particularly by releasing the tension between the need to be flexible in terms of writing specifications and the need to create consistent and verifiable documentation*.

The **quality assessment of the ICDs** seems to be another element that, regardless of any automation features that exist for this purpose, should also consider the human factor. Given the *concern among some practitioners of the risky sense of confidence these automated procedures would create*, this calls for *the inclusion of the human-in-the loop elements in this kind of documentation pipelines*. Furthermore, as criteria like 'completeness' would be context dependant, human intervention is required in extracting the required knowledge from the existing ICDs and documentation processes to configure the 'quality gates' **(quality criteria defintion)** properly. Finally, **organization policies** are key for the success of approaches like the one discussed here, by ensuring agreements about not only its adoption, but also the rigor with which the involved stakeholders will make use of it — something the feature of enforced quality criteria would contribute to.

5.2 Text-Based Descriptions withn ICDs

Unlike some of the related computer-aided approaches discussed in Section 2.3, the proposed approach combines the best of model-centered and document-centered ICD management:

the formality of models to describe critical, error-prone elements, with the flexibility of using natural language for all the other elements that such models could not describe.

In this context, it is worth discussing the viewpoints of both the TAR participants and the panel experts on the 'enforced quality' of natural language elements. On the one hand, the enforcement of writing styles is something practitioners do not feel, in principle, viable to be enforced in practice given all the workarounds that could be used to avoid it. Hence, more research would be needed on the applicability of automated style checking in the context of systems like LOFAR. This could be done by eliciting and including more tailored, domain-specific writing style rules extracted from existing documentation and practitioners' input in future cycles of the study.

On the other hand, enforcing the definition of acronyms, in addition to the integration with centralized domain/project-specific glossaries, seem to be promising. These two elements, in particular, were considered key to avoiding terminology consistency issues, which according to the expert panel are seemingly common in ICDs in multiple application domains. However, this is something worth exploring further on a study on its own, in order to improve usability for end-users, and to make proper use of this knowledge at organizational level in the same line as initiatives like *shareable glossaries*[18].

5.3 ICDs as the Single Source of Truth

The high scores on applicability and transferrability given to the *centralized index* feature highlights the importance of enabling the ICDs as the single source of truth. In particular, these evaluations show that this feature, according to practitioners and experts, is key to ensure consistency and traceability between ICDs and all the artifacts derived from them, particularly thanks to the version matching mechanism described in Section 4.4.2. However, to achieve its real potential, this feature should be adapted in a way that dependencies are tracked also at a document-element level (e.g., a particular diagram, data schemes, etc.). Furthermore, by closing the loop of this feature, that is, reporting the failed version matches back to the central hub, the organization would be able to know when outdated references are being used and take timely actions. Overall, this feature would likely help the organization decide when to move from one interface version to another, which according to the practitioners is an overall positive development.

5.4 User Experience

Finally, it is worth highlighting that the feedback given by the TAR participants on the treatment's proof-of-concept includes usability limitations. In particular, it was pointed out that setting up the writing environment was a complex process (e.g., version control system clients, CI/CD platforms, text editors, etc.). The participants mentioned that in some cases the time imposed by said limitations could exceed the time saved by its features. This, in addition to the increasing number of applications of the DaC philosophy beyond the context of software systems as described in Section 2.2, calls for further work on overcoming these user experience limitations. For instance, a fully fledged editing environment with the same extensions used by the CI/CD platform would be a valuable addition. Furthermore, in the

[18] https://thegooddocsproject.dev/docs/glossaries/

same line to the shared-glossaries initiative discussed above, more glossary management tools that allow for consistently combining and merging local glossaries with organization-level ones are worth pursuing.

6 Threats to Validity

In this section, we discuss potential threats to the validity and the actions performed to mitigate them in the study.

Construct validity deals with the degree to which the operational measures, i.e., scales, metrics and instruments, actually measure the properties under investigation, according to the research questions (Ralph and Tempero 2018). The use of two different instruments to measure the *applicability of a DaC-based ICD management approach in SoS domains* were deemed necessary to improve the validity of the study. However, each one of these instruments (*action research* and *expert panel*) are prone to threats to validity on their own. For instance, as pointed out by Petersen et al. (2014), action research — by being highly context dependent — is prone to *generalizability issues*. In particular, the results of the TAR conducted to address the first five research questions, as originally reported in Cadavid et al. (2022), were specific to ASTRON and the radio-astronomy application domain. As a means of addressing this limitation of external validity, this extended version of the study includes a new research question to explore the transferability of the result to other application domains. As described in Section 3.2.2 this research question was addressed, through the aforementioned expert panel instrument, which concluded on the generalizability of the results and identified its precise limitations. However, as LOFAR –the underlying subject of the TAR– is characterized as a *directed* SoS, it is worth noting that these results would only be generalizable to SoS with the same characteristics.

When it comes to the expert panel instrument, there is the risk that expert panel members have biases that influence their evaluation of the research. For example, they may be biased in favor of certain theories or methods. On the other hand, panel members may not have the necessary expertise to accurately evaluate the research, in this case, its transferability to other application domains. To address these limitations, the panel was made up of experts with no vested interest in the outcome of the study, and in any case participating as individuals and not as representatives of their employing organizations. Furthermore, the panel includes a group of experts which, in addition to having experience creating or working with ICDs, are diverse in terms of their respective SoS application domains and viewpoints (given their roles within said domains).

Furthermore, *researcher bias* and *selection bias* are threats to validity that any TAR study is also prone to, given particularly the involvement of the researchers in the design of the treatment under evaluation. To mitigate both, a replication package[19] is provided, including the (anonymized) qualitative data collected throughout the phases of diagnosing, action planning, and the evaluation. Likewise, the material provided to the panel members (screen casts and survey) and the qualitative data collected through the expert panel are available for scrutiny, within the same replication package.

[19] Available at https://figshare.com/s/671cc7450d8a6b8e2987

7 Conclusion

Interface management in large-scale SoS, despite being a key element in the cooperation between the disciplines involved in a SoS architecting process, has been found to also be a major pain point associated to said process (Cadavid 2021; Cadavid et al. 2022). Motivated by the potential integration and operational problems this particular pain point has been associated to, this study aimed at: (1) exploring these interface-management related issues in more depth; and (2) designing and evaluating alternative ICD management approaches to address them. To this end, in the first part of this work a *technical action research* (TAR) was conducted as a means to gather further insights on said issues, and to design and evaluate a treatment to address them, in cooperation with practitioners.

Given the document-centered engineering process used in the context in which the TAR was conducted (the LOFAR system), this treatment was based on existing approaches used in the context of software systems documentation management. In particular, the *docs-as-code* philosophy was adopted as its core, given the efficacy it has shown to address outdated or unreliable technical documentation issues in said context. This is, to the best of our understanding, the first time this philosophy has been evaluated for interface management in SoS and in complex, large systems in general. Its evaluation was twofold: a) its efficacy and fitness were evaluated by ASTRON practitioners in a single-case mechanism experiment; b) its transferability to other application domains was evaluated by a panel of experts.

Overall, the results suggest that this approach could prevent common ICD management issues in systems that already follow a document-centered approach. This is done by 1) enforcing consistency and completeness on both text-based and formal elements within the documents, and 2) turning these documents (which tend to be outdated) into an actual, traceable, single source of truth. However, like any other approach, it is not meant to be a 'silver bullet', and human factors are still key for its effectiveness.

Regarding its transferability, and according to the expert panel results, this approach is, in principle, found to be applicable across multiple domains as long as they follow a partially or fully document-centered process. The collected ICDs and the experts' viewpoint suggest that nowadays a significant number of domains follows this kind of processes over pure model-based ones — hence the importance of approaches like the one described here. This seems to be particularly the case in domains where the concept of SoS has been applied the most, arguably due to challenges and limitations of working around a single central model when the system constituents are independently developed and managed.

As future work, some areas for improvement are planned to be addressed through additional cycles in the TAR. In particular, a formalism for defining dynamic aspects of the system, an element often overlooked on the interface specifications (Cadavid 2021; Cadavid et al. 2022), would be required to also ensure that these aspects are properly incorporated on the ICDs. Furthermore, usability elements would be improved, so that the overall pipeline is easier to set up, ideally by non-developers. Finally, although the proposed approach would technically be able to manage the ICDs of SoS whose constituents have different degrees of independence (i.e., documentation distributed across multiple organizations), the TAR discussed here was focused on a *directed* SoS. More case studies on SoS with alternative authority relationships (collaborative, acknowledged, etc.). are deemed necessary to identify other potential limitation factors, such as political or privacy issues. In the same line, the generalizability of the proposed approach is expected to be further explored by conducting these studies on alternative application domains.

Data Availibility The data that support the findings of this study (referred to as *replication package* in the paper) are available in https://figshare.com/s/671cc7450d8a6b8e2987

Declarations

Conflicts of interest the authors declared that they have no conflict of interest

References

Maier MW (1998) Architecting principles for systems-of-systems. Syst Eng J Int Council Syst Eng 1(4): 267–284

ISO, IEC, (2019) 21839 information technology - systems and software engineering - system of systems (sos) considerations in life cycle stages of a system. Standard, International Organization for Standardization, Geneva, CH

Cadavid H, Andrikopoulos V, Avgeriou P (2020) Architecting systems of systems: A tertiary study. Inf Softw Technol 118:106202

Cadavid H, Andrikopoulos V, Avgeriou P, Klein J (2020) A survey on the interplay between software engineering and systems engineering during sos architecting. In Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). ESEM '20. Association for Computing Machinery, New York, NY USA

Cadavid H, Andrikopoulos V, Avgeriou P, Broekema PC (2021) System- and software-level architecting harmonization practices for systems-of-systems : An exploratory case study on a long-running large-scale scientific instrument. In 2021 IEEE 18th International Conference on Software Architecture (ICSA), pp 13–24

Cadavid H, Andrikopoulos V, Avgeriou P, Broekema PC (2022) System and software architecting harmonization practices in ultra-large-scale systems of systems: A confirmatory case study. Inf Softw Technol 150:106984

Sheard S, Creel R, Cadigan J, Marvin J, Chim L, Pafford ME (2018) Incose working group addresses system and software interfaces. INSIGHT 21(3):62–71

Cadavid H, Andrikopoulos V, Avgeriou P (2022) Documentation-as-code for interface control document management in systems of systems: A technical action research study. In European Conference on Software Architecture, pp 19–37 Springer

Gentle A (2017) Docs Like Code. Lulu Press, Inc Morrisville, North Carolina

Beck R (2015) In Lazarian A, de Gouveia Dal Pino EM. In: Melioli C (ed) Future Observations of Cosmic Magnetic Fields with LOFAR, SKA and Its Precursors. Springer, Berlin, Heidelberg, pp 3–17

Firesmith D (2010) Profiling systems using the defining characteristics of systems of systems (sos)

Thomchick R (2018) Improving access to api documentation for developers with docs-as-code-as-a-service. Proceedings of the association for information science and technology 55(1):908–910

Ozerova MI, Zhigalov IE, Vershinin VV (2020) Comparison of document generation algorithms using the docs-as-code approach and using a text editor. In Proceedings of the computational methods in systems and software, pp 315–326 Springer

Lambourne J (2017) Why we use a 'docs as code' approach for technical documentation. https://technology.blog.gov.uk/2017/08/25/why-we-use-a-docs-as-code-approach-for-technical-documentation/

Rong G, Jin Z, Zhang H, Zhang Y, Ye W, Shao D (2020) Devdocops: Enabling continuous documentation in alignment with devops. Softw Practice Exp 50(3):210–226

Rahmani K, Thomson V (2011) Managing subsystem interfaces of complex products. Int J Product Lifecycle Management 5(1):73

Louadah H, Champagne R (2014) Labiche Y. Towards automating interface control documents elaboration and management 1250:26–33

Harvey D, Waite M, Logan P, Liddy T (2012) Document the model, don't model the document. In Proc Syst Eng/Test Eval Conf 6th Asia Pac Conf Syst Eng

Karban R, Troy M, Brack GL, Dekens FG, Michaels SB, Herzig S (2018) Verifying Interfaces and generating interface control documents for the alignment and phasing subsystem of the Thirty Meter Telescope from a system model in SysML. In: Angeli GZ, Dierickx P (eds) Modeling, systems engineering, and project management for astronomy VIII, pp 29 SPIE. Austin United States

Chiozzi G, Andolfato L, Kiekebusch MJ, Kornweibel N, Schilling M, Zamparelli M (2018) Designing and managing software interfaces for the ELT. In: Guzman JC, Ibsen J (eds) Software and Cyberinfrastructure for Astronomy V, pp 78 SPIE. Austin United States

Di Maio M, Atorf L, Dahmen U, Schluse M, Rossmann J, Hoppe M, Kapos G-D (2018) Interface management with closed-loop systems engineering (CLOSE). In 2018 IEEE international systems engineering symposium (ISSE), pp 1–8

Vipavetz K, Shull TA, Infeld S, Price J (2016) Interface management for a NASA flight project using model-based systems engineering (MBSE). INCOSE Int Symposium 26(1):1129–1144

Tsui R, Davis D, Sahlin J (2018) Digital engineering models of complex systems using model-based systems engineering (mbse) from enterprise architecture (ea) to systems of systems (sos) architectures & systems development life cycle (sdlc). In INCOSE international symposium, vol 28, pp 760–776 Wiley Online Library

Japs S, Anacker H, Dumitrescu R (2021) Save: Security & safety by model-based systems engineering on the example of automotive industry. Procedia CIRP 100:187–192

Wheatcraft LS (2010) 9.2. 2 everything you wanted to know about interfaces, but were afraid to ask. In INCOSE international symposium, vol 20, pp 1132–1149 Wiley Online Library

Broy M, Böhm W, Rumpe B (2021) Advanced systems engineering. Model-Based Engineering of Collaborative Embedded Systems. Springer, New York, NY, pp 353–364

Guo D, Zhang X, Zhang J, Li H (2020) An interface management approach for civil aircraft design. In International Conference on Aerospace System Science and Engineering, pp 435–446 Springer

van Haarlem MP, Wise MW, Gunst A, Heald G, McKean JP, Hessels JW, de Bruyn AG, Nijboer R, Swinbank J, Fallows R et al (2013) Lofar: The low-frequency array. Astronomy & Astrophysics 556:2

Yasseri SF, Bahai H (2019) Interface and integration management for FPSOs. Ocean Eng 191:106441

Borrowman AJ, Taylor P (2016) Can your software engineer program your PLC? In: Chiozzi G, Guzman JC (eds) Software and Cyberinfrastructure for Astronomy IV, vol 9913, pp 99131. WA International Society for Optics and Photonics, SPIE, Bellingham

Wieringa RJ (2014) Design science methodology for information systems and software engineering. Springer, New York, NY

Kontio J, Bragge J, Lehtola L (2008) In Shull F. In: Singer J, Sjøberg DIK (eds) The Focus group method as an Empirical tool in software engineering. Springer, London, pp 93–116

Petersen K, Gencel C, Asghari N, Baca D, Betz S (2014) Action research as a model for industry-academia collaboration in the software engineering context. In Proceedings of the 2014 international workshop on long-term industrial collaboration on software engineering, pp 55–62

Fairley RE (2019) Systems engineering of software-enabled systems. Wiley Online Library, River Street Hoboken, NJ

Ralph P, Tempero E (2018) Construct validity in software engineering research and software metrics. In Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018, pp 13–23