Official URL
DOI : https://doi.org/10.1007/978-3-030-29381-9_42

Deep System Knowledge Required: Revisiting UCD Contribution in the Design of Complex Command and Control Systems

Elodie Bouzekri[1], Alexandre Canny[1], Célia Martinie[1](✉), Philippe Palanque[1,2], and Christine Gris[3]

[1] ICS-IRIT, University of Toulouse 3, Toulouse, France
{martinie,palanque}@irit.fr
[2] Department of Industrial Design, Technical University Eindhoven, Eindhoven, The Netherlands
[3] Airbus Operations SAS, Blagnac, France

Abstract. Command and control systems centralize information from multiple underlying systems to support operators in the performance of their mission. Beyond the mission itself (that may be complex), operators must also ensure the correct functioning of these systems (often called platform). Platform systems (e.g. engines or electric system) may be very different from each other and exhibit a large number of functional states. When applied to the design of command control systems, User Centered Design methods support understanding and capturing operators' needs to perform the mission, as well as to propose solutions to design usable mission-related user interfaces. However, user interfaces for platform management need to present and organize information about the underlying complex systems. Understanding those systems and abstracting away information about their behavior (so that operators can manage them) requires deep knowledge beyond UI/UX designers and UCD methods experts. In this paper, we propose a system-centered process that would complement UCD approaches for the design of command and control systems. That process takes as input the detailed functioning of underlying systems and provides abstract and structured information to inform UCD methods. Beyond supporting usability property, the integrated process supports reliability and safety properties that UCD approaches usually overlook. We present how the proposed process has been applied for the design of a large civil commercial aircraft warning system and show generalizability to other domains.

Keywords: Command and control systems · Development process · UCD · Models · Architectures

1 Introduction

User Centered Design processes [26] target the design of usable interactive systems and promote the inclusion of real users in various development phases from early needs identification and design until evaluation and deployment. Recent contributions have tried to classify and structure the various concepts underlying UCD and interaction

https://doi.org/10.1007/978-3-030-29381-9_42

design [28] as well as associating success criteria. For instance, *interaction as transmission* focusses on the information passing between the user and the system and a success criterion is the maximum efficiency for reaching a goal. Another example is *interaction as experience* which focusses on the user feelings and on subjective qualities perceived while interacting with the systems. While *interaction as transmission* could be associated to early work in Human Factors [49] (and is referred to as *classical approach* [42]), *interaction as experience* has received a lot of focus in recent HCI research building on the seminal work from Hassenzahl [25] (and is referred to as *contemporary approach*). While the classical approach was focusing on supporting users' work and avoiding the negative (such as user errors), the contemporary one mainly targets at entertainment and leisure and focusses on the positive (such as enjoyment and fun [7]). However, usability is still not trivial to reach for complex command and control systems.

This evolution might be seen as migrating from a solved problem to a new difficult problem missing clear understanding and solutions. UCD approaches are flexible but are still far from being adequate for the design and evaluation of command and control systems in general, and critical ones in particular. For instance, cockpit design by aircraft manufacturers and suppliers is performed jointly with Human Factors experts (with a deep knowledge about operators' tasks and environmental conditions) and test pilots (with a deep knowledge about missions and platform systems) [45]. This is required as command and control systems centralize information from multiple underlying systems to support operators in the performance of their mission. Beyond the mission itself (that may be complex), operators must also ensure the correct functioning of these systems (often called platform). This does not mean engaging repair activities but shutting down a faulty system or starting a redundant one [46]. The systems gathered in the platform such as a cooling system, solar panels or engines might be very different from each other an exhibit a large number of functional states very specific to each system. When applied to the design of command and control systems, User Centered Design methods support understanding and capturing operators' needs, their goals and tasks [18] in order to perform their mission. In addition, UCD approaches propose solutions to design usable user interfaces. However, when dealing with command and control that supports activities dedicated to the management of the platform, those user interfaces need to present and organize information from the underlying complex systems. Understanding those systems and abstracting away information about their behavior in order to allow operators to manage them, requires deep system knowledge beyond UI/UX designers and UCD methods experts' knowledge. Beyond, the complexity of those systems require knowledge that cannot be acquired by those UCD experts within the lifespan of the project.

As UCD approaches do not provide explicit support for building an abstract view on the system and its services, we propose a system-centered process (that would complement UCD approaches) dedicated to the design of command and control systems. That process takes as input the detailed functioning of underlying systems and provides abstract and structured information to inform the UCD of command and control systems. As UCD approaches target at improved usability, our integrated process targets at feasibility as relevant additional and required property. That process is also positioned with respect to regulations in command and control systems that target at dependability and safety. Regulatory authorities build their certification processes on top of standards that vary significantly from one domain to another. For instance, ECSS target at space systems [15], ESARR at Air Traffic Control [19] and DO-178-C at Aeronautical systems [12]. Their integration within the design processes is mandatory for critical systems and their explicit connection with UCD is thus required when building dependable, safe and usable systems [29].

The paper is organized as follows. Section 2 details the limitations of UCD approaches for command and control systems and motivations for extending UCD them with deep knowledge about underlying systems. Section 3 presents a high-level view of the proposed approach while Sect. 4 presents the foundations of the proposed process. The main steps of the System Centered Design process are presented in Sect. 5. Section 6 presents how the process has been applied for the design of a commercial aircraft warning system and Sect. 7 concludes the paper.

2 Motivations: Why UCD Is Not Enough to Design C&C Systems

The design of command and control systems requires information that is not provided by UCD (e.g. all the possible states for each device). This section highlights the actual conflicts between several principles of the UCD and the specificities of C&C systems.

2.1 Specificities of C&C Systems

The users who interact with C&C systems, named operators, interact according to predetermined procedures, predetermined tasks and predetermined behaviors [47]. Their abstract workflow [47] is to understand the system state, to compare it with the desired system state and to apply the relevant procedure and tasks to achieve the target system state. The number of possible states for devices and of information presented to the user depend on the system and devices that compose the whole system. C&C systems aim at managing large amount of system and devices, which leads to a huge number of possible operational states to deal with. It is not possible for the users know all of these possible states. In addition, the user will not be able to interact with all of the possible systems' behaviors during the whole time s/he operates the system. For example, in the case of a commercial aircraft, a fire engine may happen one time out of one billion flight hours. Most of the commercial airlines pilots' will (fortunately) never have to interact with the cockpit C&C interface to recover from an engine fire. The design and development of C&C systems are thus driven by safety and dependability objectives:

- Manufacturers of C&C systems have to set safety and dependability objectives for their systems and have to demonstrate that the delivered systems match these objectives [14]. Depending on the application domain and on the level of assurance required for a function, regulation specifications let the manufacturer proves that the targeted objective is reached using methods and techniques of its choice or indicates prescriptive means of compliance to the requirements [14]. For high assurance

levels, structured development processes and model-based approaches are means of compliance to the requirements.
- Manufacturers of C&C systems have to ensure that the C&C functions and interfaces match the users' tasks [17]. For these purposes, Human factors experts identify, gather and record exhaustively and precisely the users' tasks [16, 47].
- The design of functions and presentations for C&C tasks is performed jointly with Human Factors experts (with a deep knowledge about operators' tasks and environmental conditions) and test pilots (with a deep knowledge about missions and extant systems) [45]. The produced design documents are shared amongst the different stakeholders.

2.2 Main Principles of UCD and Their Limitations

User Centered Design is "a general term for a philosophy and methods which focus on designing for and involving users in the design of computerized systems" [1] and has the following main principles:

- Focus (early) on the user [22, 23], involve actively the user [11, 23, 31]: the user and associated characteristics, tasks and context drive the design.
  The operators have cumulative experience about the C&C systems they have been operating, and about the systems' behavior they have been interacting with. As far as they know neither all the possible system's states and all the systems' characteristics nor this information for the future systems, they do not have enough knowledge to propose design solutions that contain relevant abstraction level and relevant information. As UCD methods and techniques have historically been proposed to deal with simple in home entertainment computers [5], it may be a reason why this aspect is missing in UCD.
  Moreover, focus (early) on the systems is required too. Exhaustive and detailed information about each system is required before the design of the C&C interface. For example, in the commercial aircrafts application domain, engines are specified before the start of the cockpit design.
- Apply an iterative design process [11, 22, 31], apply an iterative and incremental system development process [23]: the design process alternates the production of multiple design solutions (evolvable prototypes) and their evaluation with users in order to accommodate requirements changes and integrate new parts of the system.
  An iterative and incremental process does not provide support to have a generic architectural view on the system and of its various components [48]. Consequently, it also makes very difficult to demonstrate that the system reach objectives in terms of levels of safety and dependability [48].
- Produce simple design representations [23]: The design solutions are represented in a way that can be easily understood by all stakeholders.
  The representations that provide support to C&C tasks have to contain the relevant information concerning the systems' characteristics and states. According to the complexity of C&C systems as well as the amount of information dealt with [47], the design representations cannot be as simple as mass-market systems design representations.

- Perform empirical measurement [22], evaluate use in context [23]: User evaluations are conducted throughout the iterative design process.
  The recruitment of operators for user evaluation as well as the preparation and implementation of the test sessions is constrained [41]. It is not possible to cover all the operators' tasks and procedures and several types of users must be involved at the same time during the test sessions.
- Multi-disciplinary design teams [23, 31].
  Several types of expertise are required (engineering, physics, human factors, software, hardware…). Stakeholders bring their expertise. The produced design and development artefacts are shared amongst the stakeholders.

2.3 Existing Approaches for Integrating UCD in System Development Processes

The fact that UCD do not explicitly address the whole development process for an interactive system has been acknowledged since decades [23, 33, 35]. From a system and software engineering perspective, development processes such as the waterfall process [43] and the V cycle process [36] have proven useful to reach safety and dependability objectives (i.e. "to build the system right") [8] but they fail in taking into account the usability property (i.e. "to build the right system") [8]. The Spiral [8] and Agile processes [44], even if they are iterative, do not explicitly take into account user needs and tasks [30]. To overcome these issues, Göransson et al. [23] proposed a design process centered on usability. Larusdottir et al. [30] proposed an approach to integrate UX design activities in Agile development processes. Gross [24] proposes a generic process based on the high level UCD phases and complemented with system development phases in order to encourage system designers and developers with no HCI background to apply UCD techniques. Martinie et al. [33] proposed a development process for safety-critical interactive systems, taking into account the development of the training program.

These existing approaches do not provide explicit guidance and support on how to use systems design artefacts within UCD phases, although system design artefacts provide the information required to identify what should be presented to the users and to take into account the feasibility of the interface and interactions design solutions. The existing approaches fail in taking into account the properties required for C&C systems: feasibility, usability, dependability and safety.

3 Holistic View on Command and Control System Development

Figure 1 presents a generic approach for taking into account the properties required for C&C systems: feasibility, usability, dependability and safety. This generic approach was designed and developed in collaboration with cockpit experts and engineers in the aeronautics domain during a four-year project on aircraft operational systems' states. This approach, named the clover process covers these properties thanks to three different sub-processes: The System Centered Design (SCD) process, the User Centered Design (UCD) process and the Regulator Centered Design (RCD) process.

The SCD process identifies all the feasible command and control system functions. Experts for each type of system should participate in this process. The aim of this process is to provide information about the available command and control devices, services and associated states by the mean of structuration and abstraction. This information is composed of data, architecture, behavioral models and sample presentation layout for the command and control system. This information thus feeds the design and the evaluation phases in the UCD process. The foundations for the information produced by the SCD process as well as the detailed phase by phase view on the process (with the documents and information that flow between the phases) is described in the next two sections of this paper.

The UCD process aims to ensure the usability property of the command and control system and should then be conducted by usability experts. HCI main principles and techniques can be applied but, as these activities require deep knowledge about the command and control system, the UCD process has to take as an input the output of the SCD process (data, models and sample presentation layouts of the systems, services and associated states). For example, the task modeling activity requires the exhaustive list of system functions, services and presentation information. If an important usability issue that is due to the C&C system is identified, the design of the C&C system must be amended (red dotted arrow from UCD to SCD in Fig. 1). In the same way, the outputs of the UCD process can be regulated during the RCD process that may output proposals for modifications (red dotted arrow from RCD to UCD in Fig. 1). System devices or services can also be adapted by SCD (green arrows propagating modifications from one process to another.

The RCD process aims to set dependability and safety properties and to verify them. It should thus be conducted by safety experts. In the aircraft domain, the DO-178C [12] standard defines development assurance level for systems of aircraft systems and associated recommendations. In addition, the CS-25 [17] defines certifications specifications and associated means of compliances for aircraft systems. Following these means of compliances, manufacturers show to the certification authorities that they developed and deployed systems that conform to these specifications and standard. A non-compliance leads to a new iteration of the UCD process or of the SCD process, and then to a new iteration of the RCD process in order to verify that the compliance issue is solved. For example, the CS-25 specifies that aircraft systems have "to be designed so that qualified flight-crew members trained in its use can safely perform their tasks associated with its intended function" [17]. Then, in order to be accepted during the RCD process, each identified user tasks (identified during the UCD process) must match a system function (defined during the SCD process with architecture and data models) for the command and control system design. This highlights the impact of dependability on usability [20].

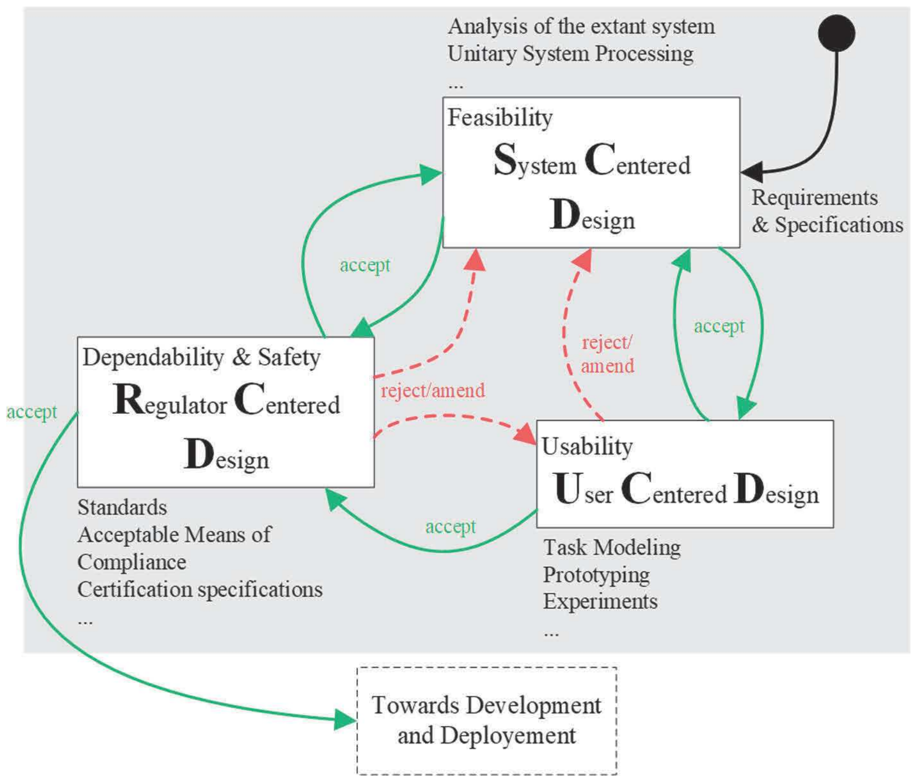
Analysis of the extant system
Unitary System Processing
...
Feasibility
System Centered Design
Requirements & Specifications
accept
reject/ amend
accept
Dependability & Safety
Regulator Centered Design
accept
reject/amend
Usability
User Centered Design
accept
Standards
Acceptable Means of Compliance
Certification specifications
...
Task Modeling
Prototyping
Experiments
...
Towards Development and Deployement

Fig. 1. The clover process for the design and development of command and control systems (Color figure online)

4 Foundations for "Systems Centered Design" for Command and Control Systems

The SCD outputs data about systems devices, services and their associated states. For example, in a commercial aircraft, the "FUEL" service is associated to the aircraft fuel systems (fuel tank, pump, cross-feed valves, etc.). In order to provide support for covering all the possible devices, services and their associated states in the design solutions for the C&C interfaces and interactions, we present: (1) an architecture built upon the concept of abstraction hierarchy framework [6], and (2) a generic and abstract state description applicable to the all of the architecture components (devices, services).

4.1 Handling Complexity with a Generic Architecture for the Command and Control of Integrated System Services and Devices

DSCU (Device, System service, Compound service and User Service) is a generic architecture designed around four types of components, each representing different level of system abstraction for command and control. Figure 2 introduces the architecture. From left to right, it goes from physical implementation of the system (devices) to services that are of interest for the end-user, named "User Service". This decomposition is close to the one proposed in the abstraction hierarchy framework [6] and thus allow to reason on the entire system.

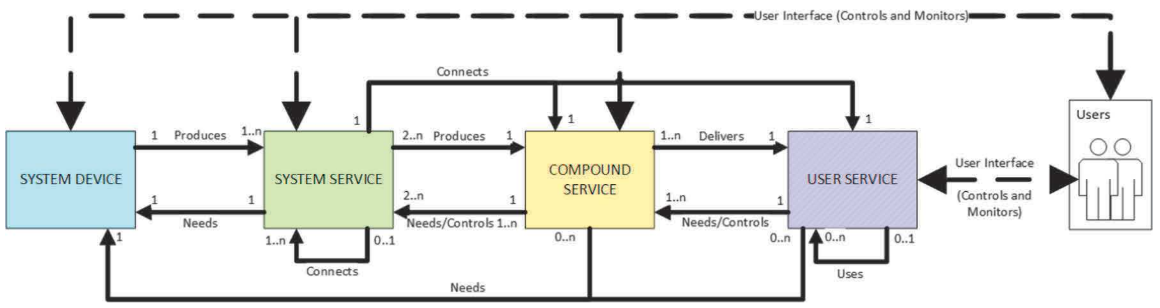

Fig. 2. Generic architecture for command and control of integrated system services and devices.

System Devices. System devices are physical devices capable of producing or routing resources (e.g. electrical generator and switches) and/or delivering forces (e.g. torque). The System Device component (leftmost box in Fig. 2) holds the operational parameter of the monitored device. On an aircraft engine, these parameters include the rotation speed of the fans, the exhaust gas temperature, etc. These are useful to identify problems such as over speed, overheating, etc.

System Service. System services are the set of resources and forces a device is capable of producing. The System Service component (second box from the left in Fig. 2) holds information regarding the production of the said resource or force, named "Service" for generalization purpose. The System Services produced by an aircraft engine are, for example, "Thrust" or "AC Electricity". The System Service "Engine 1 AC Electricity" holds information related to the monitoring of the Engine 1 generator such as output Voltage or Current.

Devices dedicated to routing (e.g. electrical switches, fuel valves) enable system services such as "Electricity routing" or "Fuel routing".

Compound Service. A Compound Service is a system-wide resource made available to devices and other services after aggregation from multiple producers. For example, on a twin-engine aircraft, the "AC Electricity" compound service is the result of the compounding of "Engine 1 AC Electricity", "Engine 2 AC Electricity", "Auxiliary Power Unit AC Electricity" and "AC Electricity Routing". "Compound Service" (third box for the left in Fig. 2) are particular as their operational parameters may be nominal even though some system services used to produce it are faulty. Indeed, in complex systems, redundancy is an example of safety mechanism designed to prevent complete loss of compound services. This means that even though a system service is not working anymore (e.g. "Engine 1 AC Electricity"), the compound service to which it participates (i.e. "AC Electricity") may still be properly made available. The role of the "Compound Service" is to allow for the identification of such combination and their proper monitoring.

User Service. A User Service (fourth box from the left in Fig. 2) is a service that is of interest for the user, or in other word directly associated to his/her goal. It needs and controls one or multiple "Compound Services" in order to be delivered. For example, on an aircraft preparing for takeoff at night, the flight crew needs to dim the cabin light in order to comply with safety procedure. In this case, "Cabin Light" is a user service that principally relies on the “AC Electricity” compound service in order to be delivered.

4.2 Handling State Explosion with a Generic and Abstract Systems and Services States Description

OQCR is a generic state based framework designed to allow the description of the status of devices and systems according to 4 variables. These variables were identified by analyzing the existing synoptic pages and alerts on command and control systems. The variables used in OQCR are:

- **Operational State**: Is the device/service on, off, powering on, etc.?
- **Qualitative State**: Is the device/service working properly? (e.g. is a battery delivering sufficient or insufficient voltage? Is it dead?)
- **Contextual Attribute**: Is the device/service in a suitable environment for operating properly? (e.g. is it within operating temperature range? Is it plugged to a suitable electrical network (voltage, frequency)?
- **Restrictive Attribute**: Is it allowed to use the device/service?

The values these variables may receive are either boolean (for the attributes) or extracted from a set of component-dependent values (for the states). The second line of Table 1 details the size of each set of values the OQCR variables (first line in Table 1) may receive. This section presents (i) the sets of OQCR states and (ii) the sets of OQCR attributes.

Table 1. Structure of an OQCR state/attributes description.

Operational state	**Q**ualitative state	**C**ontextual attribute	**R**estrictive attribute
1 value out of 4	1 value out 3	1 out of 2	1 out of 2

Operational and Qualitative States. In OQCR, the Operational and Qualitative states are meant to provide real-time and predictive information regarding the behavior of the devices and services. To do so, they indicate whether the device/service is in operation (Operational state) and to which extant it is operating/it can operate properly (Qualitative State). Since DSCU covers a variety of components, slight variations in the wording of the state values help to reflect the role of each component. Table 2 presents the Operational states of OQCR.

Table 2. OQCR Operational states for components of the DSCU architecture

Device	System service	Compound service	User service
NOT RUNNING	NOT PRODUCING	NOT DELIVERING	
STARTING	RAMPING UP		
RUNNING	PRODUCING	DELIVERING	
SHUTING DOWN	RAMPING DOWN		

The values for the qualitative attributes are presented in Table 3. While the Operational state is a real-time value only, we observe that the Qualitative state owns a predictive value when a device/service is not running/producing/delivering. Indeed, it may indicates that attempting the use the device/service will lead to either (1) the expected behavior, (2) an unexpected behavior or (3) a failure to start due to a previously identified loss of the device/service.

Table 3. OQCR Qualitative states for components of the DSCU architecture

Device	Services	Definition
FUNCTIONAL		The device can run or run properly. The service is (or can be) produced/delivered as required
DEGRADED		The device is not capable of running properly and suffers performance penalty. The service cannot be produced or delivered as required
OUT OF ORDER	OUT OF SERVICE	The device is not capable to run. The service cannot be produced or delivered

Contextual and Restrictive Attributes. The OQCR Contextual and Restrictive attributes are meant to provide information regarding the environment the device/service evolves in (Contextual attribute) and how safe it is to use it (Restrictive attribute). Table 4 presents and define the Contextual and the Restrictive attributes for DSCU components. It is important to note that multiple factors such as resource availability (e.g. low fuel pressure for an engine) or environment-related ranges (e.g. temperature range, altitude range) impact the contextual attribute. The restrictive attribute is the result of the computation of other system state that may forbid the usage of a given device/service under some circumstances. (e.g. if a ventilation system is **out of service**, it is **not allowed** to use the device it is meant to cool).

Table 4. OQCR attributes for components of the DSCU architecture.

Attribute	Value	Definition
Contextual	WITHIN CONTEXT	The device/service is in its nominal context of use
	OUT OF CONTEXT	The device/service is not in its nominal context of use
Restrictive	ALLOWED	The device/service can be use
	NOT ALLOWED	The device or service must not be in use

5 A Detailed Process for "System Centered Design" of Command and Control Systems

The SCD process aims at structuring and abstracting the C&C systems' descriptions and states. This process uses the DSCU architecture to describe each system in a structured way and the OQCR states abstraction for each component of the systems described with the DSCU architecture. The outputs of this process are an integrated architecture of the systems, the behavioral models of the C&C system and sample presentation layouts of devices and services states. HCI designers can use these artefacts during the UCD process to propose design solutions for the C&C user interface. Figure 3 presents the System Centered Design process of C&C systems. The process takes as input all the systems being under supervision of the extant command and control system. The process steps are applied for each system one by one and the last step consists in integrating the information produced for each system.

5.1 Data Collection

This step consists in collecting the extant information concerning the selected system. Each system must have a well-specified documentation including specification of alarms, services, and operation and training manuals.

5.2 DSCU Generic Architecture and OCQR States Instantiation

During the second step, system designers analyze the selected system according to the DSCU architecture. The designers have to identify the devices, the routing devices, the system services, the compound services and the user services composing the selected system. The specification documentation on the services of the system and the operation and training manuals are used during this step.

This steps aims at identifying if a service is useful to enable another service or system, and if a service is directly useful to reach a user goal. For example, in the case of a commercial aircraft cockpit, if the pilot needs to perform a “climb” at the beginning of the flight, then the service “climb” is a User Service. The operation and training manual are helpful to understand the services utility for users’ tasks. The result of this analysis is an instantiated DSCU architecture. The OQCR states provide an abstraction of all the possible values of devices or services parameters (e.g. value of speed or quantity) for this architecture. However, in some particular domain, specific parameters may be important to abstract the device or service state. During this step, the OQCR states (presented from Sect. 4.2) can thus be customized for a particular device or service if needed for the application domain (an example of such customization is presented in Sect. 6.1).

5.3 Unitary System Processing

The aim of this step is to detail the behaviour of each component of the instantiated DSCU architecture. The description of the devices and services behaviour will be useful during the UCD process to understand which events trigger a state change. In order to achieve this, the designer must follow three sub steps described hereinafter.

Unitary Alarms Identification. This sub step aims to assign each alarms of the system to the DSCU components.

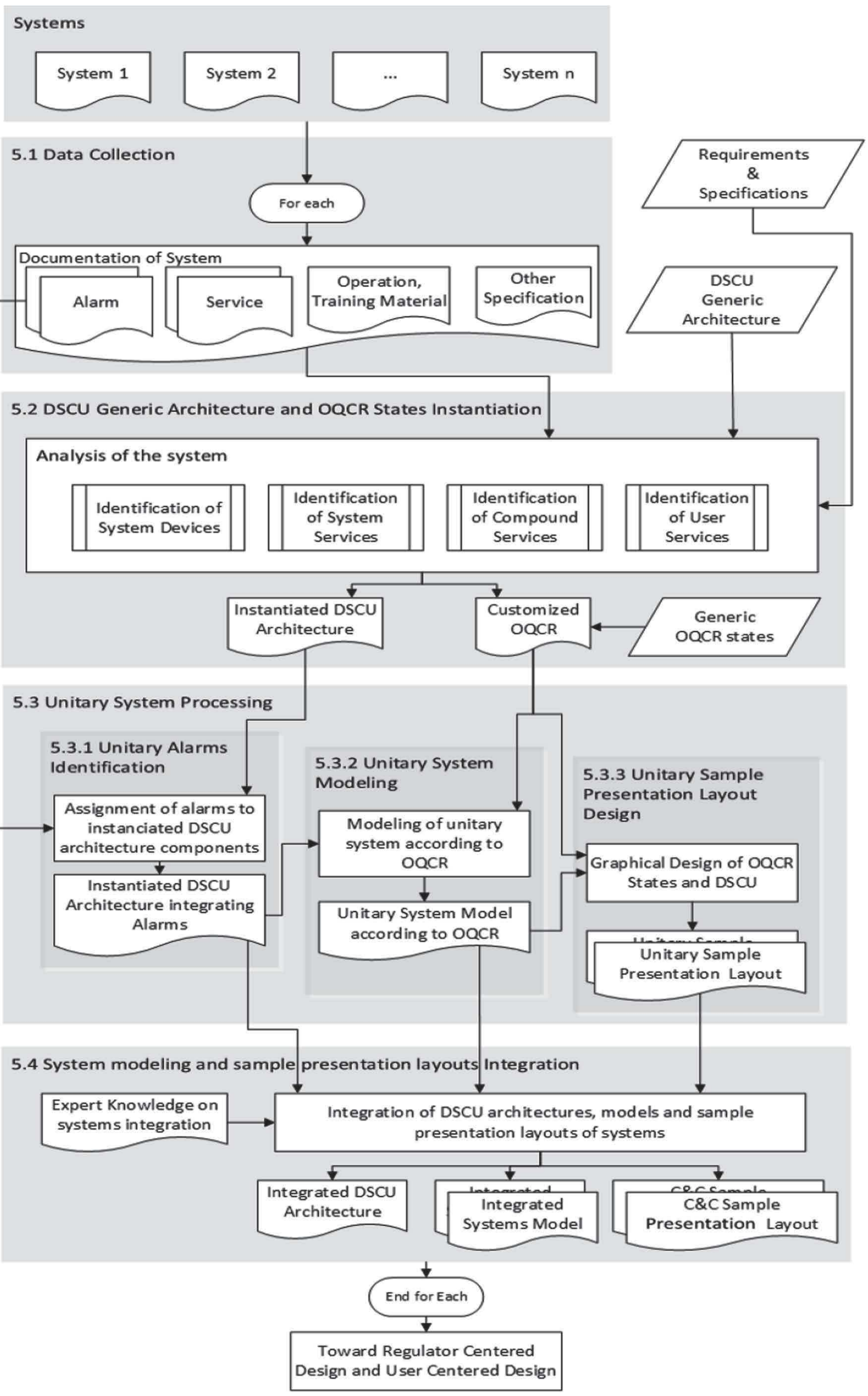
Systems
System 1
System 2
...
System n
5.1 Data Collection
For each
Documentation of System
Alarm
Service
Operation, Training Material
Other Specification
Requirements & Specifications
DSCU Generic Architecture
5.2 DSCU Generic Architecture and OQCR States Instantiation
Analysis of the system
Identification of System Devices
Identification of System Services
Identification of Compound Services
Identification of User Services
Instantiated DSCU Architecture
Customized OQCR
Generic OQCR states
5.3 Unitary System Processing
5.3.1 Unitary Alarms Identification
Assignment of alarms to instanciated DSCU architecture components
Instantiated DSCU Architecture integrating Alarms
5.3.2 Unitary System Modeling
Modeling of unitary system according to OQCR
Unitary System Model according to OQCR
5.3.3 Unitary Sample Presentation Layout Design
Graphical Design of OQCR States and DSCU
Unitary Sample Presentation Layout
5.4 System modeling and sample presentation layouts Integration
Expert Knowledge on systems integration
Integration of DSCU architectures, models and sample presentation layouts of systems
Integrated DSCU Architecture
Integrated Systems Model
C&C Sample Presentation Layout
End for Each
Toward Regulator Centered Design and User Centered Design

Fig. 3. System centered design of command and control systems

This assignment is the answer to the question “is the alarm affecting the system device, routing device (or service), system service, compound service or user service?” This step of alarm assignation is useful to determine in which conditions the device or service under consideration is “DEGRADED”, “OUT OF ORDER/SERVICE”, “OUT OF CONTEXT” or “NOT ALLOWED”.

Unitary System Modeling. This step consists in producing a description of the behavior of the system following the DSCU instantiated architecture with alarms and OQCR states in this sub step. The behavior of the device or service can be described thank to various languages and notations like automata, Petri nets, or flow-based notation. This model describes in which conditions and after which events the device or service under consideration changes state.

Unitary Sample Presentation Layout Design. This step consists in producing sample presentation layouts to help to understand the available services of the system and the behavior of the system according to the alarms and user tasks. The sample layouts must make visible all that is feasible with the system under consideration, as well as all the possible states for the presented system components. Several different sample presentation layouts of the C&C interface for the system are produced in accordance with operational scenarios. These different sample presentations will then be easier to use during the UCD process as they are explicitly bound to scenarios. The operations and training manuals contain procedures for nominal and abnormal situations taking into account the system context. This is why they are useful resources for this sub step.

5.4 System Modeling and Sample Presentation Layouts Integration

The C&C interface must provide an integrated vision of the systems’ characteristics and states. The last step of the SCD process is thus to integrate the outcome of the previous steps for each processed system. The integration of the information about the different systems can reveal new services or some introduced errors during the structuration and abstraction process. Then, the expertise of the C&C systems’ experts is needed to correctly identify the final services of the integrated systems. This integration step produces the final integrated DSCU architecture, the final integrated systems models and the finals systems sample presentation layouts when every system of the whole command and control system were processed.

6 Application of the System Centered Design Process to an Interactive Cockpit Application

In this section, we present a summary of the result we obtained applying the SCD process to the design of a future crew alerting system for large civil aircrafts. This work was performed in collaboration with Airbus Operation SAS in a project called “Integration of the Cockpit and its Systems”. This project involved, at various level, additional stakeholders such as Airbus Helicopters, Dassault Aviation and Thales Avionics.

The commercial aircraft Airbus A350-900 is the system selected for the case study. The following set of materials and documentation were used for the application of the process:

- **User-related materials** such as the aircraft Flight Deck Briefing for Pilots and its Flight Crew Operation Manual (FCOM);
- **Training materials** such as the Flight Crew Training Manual (FCTM);
- **Specifications of aircrafts systems** such as logical datasheets for the Warning System, system specifications, system requirements, etc.;
- **Regulatory documentation** including standard for software development in aeronautics [12], design assurance guidance for airborne systems [13] and Certification Specification and Acceptable Means of Compliance for large aeroplanes [10].

During the first quarter of the project, weekly meetings with a Cockpit Display expert, and a Flight Warning Engineer were dedicated to the analysis of the input materials and documentations. The Flight Warning System (FWS) aggregates data from most aircraft systems, hence its development in close cooperation with experts of aircraft systems providing input to the FWS. The next two months focused on a subset of the aircraft systems: Auxiliary Power Unit (APU), Bleed, Electricity, Fire Protection System (FPS) and Fuel. For each of these systems, we collected data from the documentation (5.1) and used this data to derive an architecture from DSCU (5.2). At this point, we realized that the wording of OQCR states and attributes could be refined for each components of the architecture (5.2). A Flight Warning System expert then joined the project to help us with step 5.3. The Flight Warning System expert contributed largely to the integration step (5.4). We had several meetings with various stakeholders to validate the outcome of each step. After each review, we recorded a list of modifications and amendments of the architectures, models and presentation layouts. We then validated the new versions of the architectures, the models and the presentation layouts with engineers and experts before pursuing the project at a larger scale. This section presents an extract of the application of the SCD process to the aircraft system: Engines.

6.1 Example of the Application of the SCD Process to the Aircraft Engines

Data Collection (step 5.1 in the Fig. 3). The documents gathered to understand the functioning and the use of aircraft engines are:

- the specification of the engines (sections concerning the behaviour, envelope, services and needed resources) in the FCOM (Flight Crew Operation Manual) [2],
- the list of possible alarms for the engines and their associated recovery procedures (FCOM too),
- the usage instructions with associated C&C interfaces screenshots in the FCOM and the FCTM (Flight Crew Training Manual).

A screenshot of a possible state of the extant Command and Control (C&C) interface for engines in an A350-900 cockpit is presented in Fig. 4. The information presented is related to "ALL ENG FLAME OUT" (both engines stopped running) alarm. This status of the engines can be derived from the display as:

1. the engines are indicated as failed (represented with the amber attention getting boxes in left-hand side of Fig. 4) and
2. the vibrations of rotors are not updated anymore (represented with the "XX" amber indications for the vibrations of rotors).

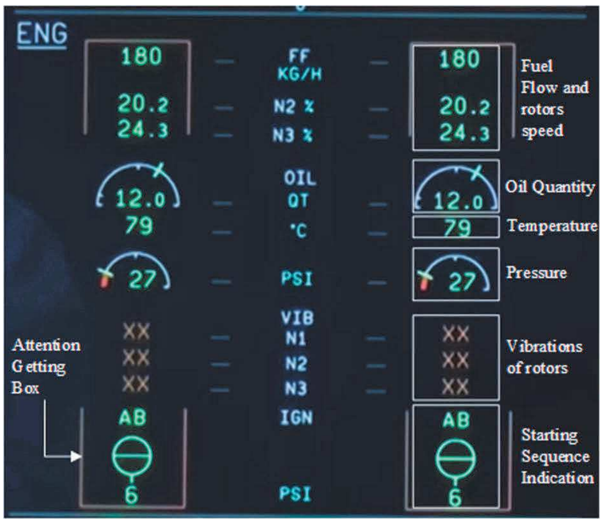

Fig. 4. Current ENGines System Display page after "ALL ENG FLAME OUT" alarm triggered

DSCU Generic Architecture and OQCR States Instantiation. (step 5.2 in Fig. 3)

DSCU Instantiated Architecture for Engines. Figure 5 presents the DSCU architecture instantiated for the two engines of the Airbus A350-900. It includes two ENGines System Devices (blue components of Fig. 5 labelled "ENG 1" and "ENG 2"). From the specification of the engines, we know that both engines produce HYDRaulic, BLEED, ELECtricity and THRUST (i.e. force pushing forward) services. Then, both ENGines produce a System Service for each of these services (green components connected to ENG 1 and ENG 2 in Fig. 5 labelled "ENG 1 HYDR", "ENG 1 BLEED"…). Combined, ENG 1 and ENG 2 produce services corresponding to the merging of the services of each Engine. These Compound Services (yellow components connected to engines System Services in Fig. 5) include "THRUST" from "ENG 1 THRUST" and "ENG 2 THRUST", BLEED … Each service requires a routing device and a routing service to be transported. In consequence, the DSCU architecture of the engines includes a routing System Service for each Compound Services (components labelled routing and network System Service components of Fig. 5: "ELEC NETWORK", "BLEED ROUTING"…).

In addition, the engines need a FUEL service for their operation. Then, the DSCU architecture of engines includes a FUEL Compound Service as a resource for ENG to function (FUEL Compound Service yellow component on the left-hand side of Fig. 5).

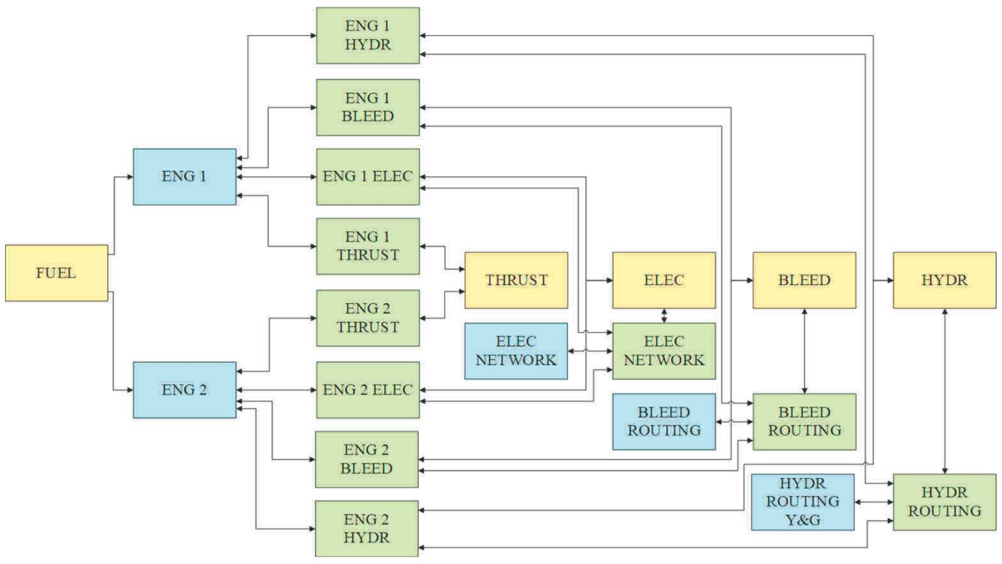

Fig. 5. Instantiated DSCU architecture for engines (Color figure online)

OQCR States Customization. During the analysis process, we found that the engines may be in an unknown state when they have not been turned on yet. Indeed, we cannot know its quality of operation state. In the same way, in case of a data update failure from the systems to the C&C interface, the whole state of the system is unknown. In consequence, we customized OQCR description by adding a value “UNKNOWN” that applies to each OQCR state descriptor (O and Q) and each attribute (C and R).

Unitary System Processing. (step 5.3 in Fig. 3)

Unitary Alarms Identification. The ENGines documentation contains all the alarms that may occur with this system. The FCOM indicates 60 possible alarms for the engines [2]. We present here two examples assignment of alarms to the components of the DSCU architecture: the caution alarm “THRUST LOCKED” and the warning alarm “ALL ENG FLAME OUT”. The “THRUST LOCKED” alarm indicates that the THRUST is frozen for one or both ENGines. The concerned components in DSCU are ENGines THRUSTs (green ENG 1 TRHUST and ENG 2 THRUST System Services components in Fig. 5). The “ALL ENG FLAME OUT” alarm indicates that both engines are shutdown during the flight, represented in DSCU by ENGines devices (blue ENG 1 and ENG 2 System Devices components in Fig. 5).

Unitary System Modeling. We used ICO Petri nets [39] to describe the behaviour of the engines. One of the reasons we choose ICO notation is its ability to scale that is needed for the integration step. Beyond, this notation has been widely used in multiple domain for describing interactive systems behaviours for cockpits [9] or Air Traffic Control Workstations [40]. As it is grounded on Petri nets theory, this notation is also able to deal with concurrency and large number of states, beyond what State machines can represent.

Unitary Sample Presentation Layout Design. To produce sample presentation layout of engines states, we used the recommended recovery actions (for abnormal situations) and normal checklists (for nominal situation) described in the FCOM [2].

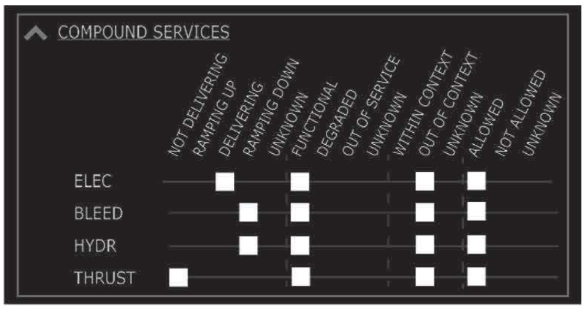

Fig. 6. A sample presentation layout of DSCU structuration and the OQCR states of the engines Compound Services in case of ALL ENG FLAME OUT.

Figure 6 presents one mockup of a presentation layout. It depicts the states of every Compound Services of the engines when a "ALL ENG FLAME OUT" warning alarm occurs. The mockup shows that, after the occurrence of that alarm, the ENGines System Devices are in the operational state "NOT RUNNING". All the Compound Services are "OUT OF CONTEXT" (abnormal lack of resources). In addition, BLEED and HYDraulic Compound Services are "RAMPING DOWN" as the ENGines stop producing this services. The user service THRUST is "NOT DELIVERING" and the engines are "NOT RUNNING". The ELECtricity Compound Service is still "DELIVERING" thanks to an automatically turned on backup system for ELECtricity. Finally, all the Compound Services are still "FUNCTIONAL" and have the restrictive attribute "ALLOWED" because the warning concerns the ENGines System Devices and that there are no restriction of usage in this context for these Compound Services.

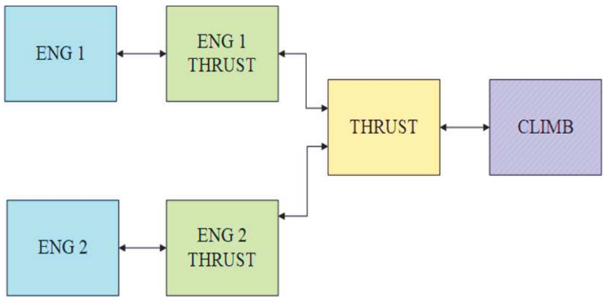

Fig. 7. Integration of CLIMB user service in the DSCU architecture of engines

System Modeling and Integrated Presentation Layouts. (step 5.4 in the Fig. 3)

Integrated DSCU Architecture. Together with the experts, we identified that THRUST Compound Service produces the CLIMB User Service (used by the pilots during takeoff for example). We thus integrated CLIMB User Service into the DSCU architecture as shown in Fig. 7.

Integrated Systems Models. During this sub step, we connected the model of the behavior of the THRUST Compound Service to the model of the behavior of CLIMB User Service. These behavioral models are not presented here due to space constraints. The interested reader can find similar models (covering nominal and abnormal situations in [21]).

C&C Sample Presentation Layout. Figure 8 presents a mockup of presentation layouts produced for the "ALL ENGs FLAME OUT" alarm. In this layout, the ENG 1 and ENG 2 System Devices are in the qualitative state DEGRADED because of the alarm.

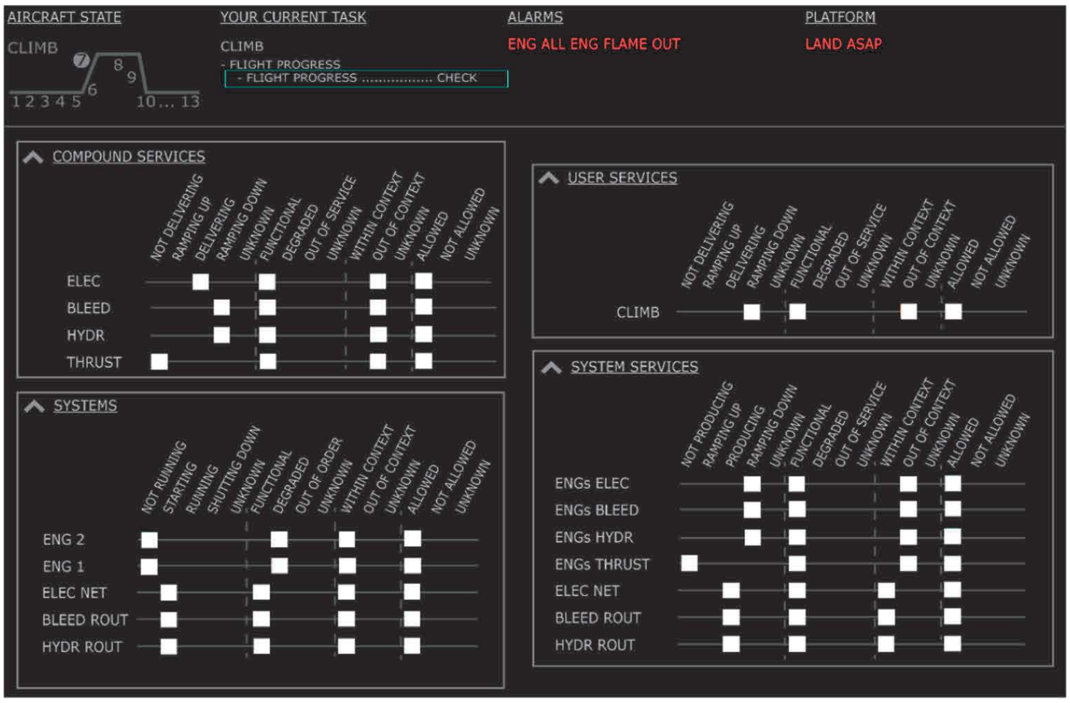

Fig. 8. A layout of the presentation for the system under design (following DSCU and OCQR).

In addition, they are in the operational state NOT RUNNING because the warning alarm indicates that they are shutdown. All of the System Services related to the ENGines ("ENGs ELEC", "ENGs BLEED", "ENGs HYDR" and "ENGs THRUST") have the contextual attribute OUT OF CONTEXT and they are in the operational state RAMPING DOWN, except the THRUST System Service that is in the operational state NOT PRODUCING. The CLIMB User Service is in the operational state RAMPING DOWN and has the contextual attribute OUT OF CONTEXT. Indeed, the THRUST resource is no longer produced and the plane will slowly start to glide. Routing System Services are still PRODUCING (operational state) –FUNCTIONAL (qualitative state) -WITHIN CONTEXT (contextual attribute) –ALLOWED (restrictive attribute).

6.2 Towards UCD of the Crew Alerting System

Each artefact produced during the application of the SCD process can be exploited by most UCD techniques and methods. The set of DSCU architectures and the OQCR (states and attributes) provides the exhaustive list of devices, services and their associated states and attributes. This information provides support for designing mock-up and prototypes of the interface of the C&C interactive systems. They also provide support to crosscheck the prototypes and the task models, in order to determine whether all the devices are bound to at least one user task. They also can be of great help provide support to observation and interview activities as they help understanding which services and devices are required for each user service (e.g. the connection between the engines and climb in DSCU architecture represents explicitly the fact that engines have to be functional to perform CLIMB). The system components behavioral models provide additional support to define the behavior of mockups and prototypes beyond their layouts. In the same ways, the coverage of both nominal and abnormal situations provide support to produce prototypes that cover all the cases but also helps identifying in an exhaustive way operators' tasks [37] identifying corner cases to be addressed in interviews and observations preparation. Furthermore, the system components behavioral models provide support to the application of dependable computing techniques [38], that provides means of compliance for the application of the RCD process.

7 Conclusion and Perspectives

While UCD approaches focus on the usability of the interactive system under design, constraints beyond users' needs that have to be taken into account when designing complex command and control systems. For instance, feasibility is a first class citizen but it is not addressed by UCD approaches. For instance, without a deep understanding of aircrafts physics, it is not possible to design a cockpit that will meet feasibility. Beyond, this paper has demonstrated that without a deep understanding of aircraft systems, cockpit design is a task doomed to fail, especially when interfaces for system management are concerned. However, it is not possible for designers to learn all this information for every type of command and control system they are likely to design. This paper tackles that specific problem by providing a generic design process for gathering the information of underlying systems when designing a command and control system. This process makes explicit use of available documentation (both technical and operational) and provides stepwise progress towards User Centered Design.

In this paper, the proposed approach is applied to the design of user interfaces for aircraft cockpits. However, the approach is generic enough and applicable to other command and control systems. For example, in the space domain, the satellite platforms and the missions they support are firstly designed for feasibility. The design of ground segment applications to monitor and control the various sets of devices of the satellites and of the ground communications systems also requires knowledge beyond the UI/UX designers and UCD experts' knowledge [32]. The paper has also emphasized the importance of standards and certification activities in the design of these systems. Indeed, even automotive systems rely on existing standards such as AUTOSAR [4] and ISO 26262 [27] that provide regulatory framework for autonomous cars design and development. We believe that this paper can provide support to designers involved in design tasks of C&C systems that have been so far not supported by UCD processes leaving, very unfortunately, Command and Control interactive systems design incapable to benefit from UCD benefits. Such approach requires techniques to trace and analyze the coverage of regulatory requirements, feasibility requirements and design options [34].

References

1. Abras, C., Maloney-Krichmar, D., Preece, J.: User-centered design. In: Bainbridge, W. (ed.) Encyclopedia of Human-Computer Interaction. Sage Publications, Thousand Oaks (2004)
2. Airbus A350 Flight Crew Operating Manual, 5T1 A350 FLEET FCOM. Technical Report. Airbus
3. APA 2017: Publication Manual of the American Psychological Association, 6th edn
4. AUTOSAR AUTomotive Open System ARchitecture development: “Foundation”. www.autosar.org. Accessed 14 May 2018
5. Beaudouin-Lafon, M.: Designing interaction not interfaces. In: Proceedings of AVI, pp. 15–22. ACM (2004)
6. Bisantz, A.M., Vicente, K.J.: Making the abstraction hierarchy concrete. Int. J. Hum Comput Stud. **40**, 83–117 (1994)
7. Blythe, M., Monk, A. (eds.): Funology 2. HIS. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-68213-6
8. Boehm, B.: A spiral model of software development and enhancement. ACM SIGSOFT Softw. Eng. Notes **11**(4), 14–24 (1986)
9. Bouzekri, E., et al.: Engineering issues related to the development of a recommender system in a critical context: application to interactive cockpits. Int. J. Hum Comput Stud. **121**, 122–141 (2019)
10. CS-25 – Amendment 17 - Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes. EASA (2015)
11. Dix, A., Finlay, J., Abowd, G., Beale, R.: Human-Computer Interaction, 3rd edn. (2004)
12. DO-178C/ED-12C: Software Considerations in Airborne Systems and Equipment Certification. RTCA and EUROCAE (2012)
13. DO-254/ED-80: Design Assurance Guidance for Airborne Electronic Hardware, published by RTCA and EUROCAE (2000)
14. Dodd, I., Habli, I.: Safety certification of airborne software: an empirical study. Reliab. Eng. Syst. Saf. **98**(1), 7–23 (2012)
15. ECSS-Q-ST-40C Safety, March 6, 2009, Safety Standards for Space Systems, European Space Agency. European Cooperation for Space Standardization
16. Endsley, M.R.: Designing for Situation Awareness: An Approach to User-Centered Design, 2nd edn. CRC Press Inc., Boca Raton (2011)
17. European Aviation Safety Agency: CS-25 – Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes (2017)
18. Eurocontrol Model for Task and Job Descriptions of Air Traffic Controllers, HUM.ET1.ST01.1000-REP-01. EATCHIP Reference Material (1996)
19. ESARR 4 - Risk Assessment and Mitigation in ATM & Acceptable Means of Compliance with ESARR 4, Oct 2009, Eurocontrol

20. Fayollas, C., Martinie, C., Palanque, P., Deleris, Y., Fabre, J.-C., Navarre, D.: An approach for assessing the impact of dependability on usability: application to interactive cockpits. In: EDCC 2014, Newcastle, pp. 198–209 (2014)
21. Fayollas, C., Palanque, P., Fabre, J.-C., Martinie, C., Déléris, Y.: Dealing with faults during operations: beyond classical use of formal methods. In: Weyers, B., Bowen, J., Dix, A., Palanque, P. (eds.) The Handbook of Formal Methods in Human-Computer Interaction. HIS, pp. 549–575. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51838-1_20
22. Gould, I.D., Lewis, C.: Designing for usability: key principles and what designers think. Commun. ACM **28**(3), 300–311 (1985)
23. Göransson, B., Gulliksen, J., Boivie, I.: The usability design process – integrating user-centered systems design in the software development process. In: Software Process Improvement and Practice, vol. 8, no. 2, pp. 111–131. Wiley (2003)
24. Gross, T.: *UCProMo*—towards a user-centred process model. In: Bogdan, C., et al. (eds.) HCSE/HESSD -2016. LNCS, vol. 9856, pp. 301–313. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44902-9_19
25. Hassenzahl, M., Platz, A., Burmester, M., Lehner, K.: Hedonic and ergonomic quality aspects determine a software's appeal. In: ACM CHI Conference, pp. 201–208 (2000)
26. ISO 9241-210: Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems, Geneva
27. ISO 26262, Road vehicles – Functional safety (2011)
28. Hornbæk, K., Oulasvirta, A.: What is interaction? In: Proceedings of the CHI Conference on Human Factors in Computing Systems, pp. 5040–5052 (2017)
29. Ladry, J.-F., Navarre, D., Palanque, P.: Formal description techniques to support the design, construction and evaluation of fusion engines for sure (safe, usable, reliable and evolvable) multimodal interfaces. In: ACM International Conference on Multimodal Interaction, pp. 185–192 (2009)
30. Larusdottir, M., Gulliksen, J., Cajander, A.: A license to kill – improving UCSD in Agile development. J. Syst. Softw. **123**, 214–222 (2017)
31. Maguire, M.: Methods to support human-centred design. Int. J. Hum Comput Stud. **55**(4), 587–634 (2001)
32. Martinie, C., Palanque, P., Fahssi, R., Blanquart, J.-P., Fayollas, C., Seguin, C.: Task model-based systematic analysis of both system failures and human errors. IEEE Trans. Hum. Mach. Syst. **46**(2), 243–254 (2016)
33. Martinie, C., Palanque, P., Navarre, D., Barboni, E.: A development process for usable large scale interactive critical systems: application to satellite ground segments. In: Winckler, M., Forbrig, P., Bernhaupt, R. (eds.) HCSE 2012. LNCS, vol. 7623, pp. 72–93. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34347-6_5
34. Martinie, C., Palanque, P., Winckler, M., Conversy, S.: DREAMER: a design rationale environment for argumentation, modeling and engineering requirements. In: Proceedings of the 28th ACM International Conference on SIGDOC, pp. 73–80. ACM, New York (2010)
35. Mayhew, D.J.: The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design, 1st edn. Morgan Kaufmann Publishers Inc., San Francisco (1999)
36. McDermid, J., Ripken, K.: Life cycle support in the Ada environment. SIGAda Letters (1983)
37. Martinie, C., et al.: Formal tasks and systems models as a tool for specifying and assessing automation designs. In: Proceedings of the International Conference on ATACCS, Toulouse, pp. 50–59 (2011)

38. Navarre, D., Palanque, P., Basnyat, S.: A formal approach for user interaction reconfiguration of safety critical interactive systems. In: Harrison, M.D., Sujan, M.-A. (eds.) SAFECOMP 2008. LNCS, vol. 5219, pp. 373–386. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87698-4_31
39. Navarre, D., Palanque, P., Ladry, J.-F., Barboni, E.: ICOs: a model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. ACM Trans. Comput.-Hum. Interact. **16**, 18:1–18:56 (2009)
40. Navarre, D., Palanque, P., Bastide, R.: A tool-supported design framework for safety critical interactive systems. Interact. Comput. **15**(3), 309–328 (2003)
41. Reuzeau, F.: Finding the best users to involve in design: a rational approach. Le travail humain **64**(3), 223–245 (2001)
42. Rogers, Y.: HCI theory: classical, modern, and contemporary. Synth. Lect. Hum.-Centered Inform. **5**(2), 1–129 (2012)
43. Royce, W.: Managing the Development of Large Software Systems, pp. 1–9. IEEE Wescon (1970)
44. Schwaber, K.: Agile Project Management with Scrum. Microsoft Press (2004)
45. Singer, G.: Minimizing pilot-error by design: are test pilot doing a good enough job? Hum. Factors Aerosp. Saf. **1**(4), 301–321 (2001)
46. Singer, G., Dekker, S.: Pilot performance during multiple failures: an empirical study of different warning systems. Transp. Hum. Factors **2**(1), 63–76 (2000)
47. Stanton, N., et al.: Development of a generic activities model of command and control. Cog., Tech. & Work (2008)
48. Turk, D., France, R., Rumpe, B.: Limitations of agile software processes. In: Proceedings of the International Conference on eXtreme Programming and Agile Processes in Software Engineering, Italy (2002)
49. Vicente, K.: Cognitive Work Analysis: Toward Safe, Productive, and Healthy Computer-Based Work, 417 p. Lawrence Erlbaum Associates (1999)