



**HAL**  
open science

# Should I Add Recommendations to My Warning System? The RCRAFT Framework Can Answer This and Other Questions About Supporting the Assessment of Automation Designs

Elodie Bouzekri, Célia Martinie, Philippe Palanque, Katrina Atwood,  
Christine Gris

## ► To cite this version:

Elodie Bouzekri, Célia Martinie, Philippe Palanque, Katrina Atwood, Christine Gris. Should I Add Recommendations to My Warning System? The RCRAFT Framework Can Answer This and Other Questions About Supporting the Assessment of Automation Designs. 18th IFIP Conference on Human-Computer Interaction (INTERACT 2021), IFIP Technical Committee 13 on Human-Computer Interaction, Aug 2021, Bari, Italy. pp.405-429, 10.1007/978-3-030-85610-6\_24 . hal-03376253

**HAL Id: hal-03376253**

**<https://hal.science/hal-03376253>**

Submitted on 25 Sep 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License



This document is the original author manuscript of a paper submitted to an IFIP conference proceedings or other IFIP publication by Springer Nature. As such, there may be some differences in the official published version of the paper. Such differences, if any, are usually due to reformatting during preparation for publication or minor corrections made by the author(s) during final proofreading of the publication manuscript.

# Should I add Recommendations to my Warning System? The RCRAFT Framework can Answer This and Other Questions about Supporting the Assessment of Automation Designs

Elodie Bouzekri<sup>1</sup>, Célia Martinie<sup>1</sup>, Phil Palanque<sup>1</sup>, Katrina Atwood<sup>2</sup>, Christine Gris<sup>3</sup>

<sup>1</sup> ICS-IRIT, Université Toulouse III-Paul Sabatier, Toulouse, France

<sup>2</sup> University of York, York, United Kingdom

<sup>3</sup> Airbus Operations SAS, Blagnac, France

{elodie.bouzekri, celia.martinie, philippe.palanque}@irit.fr  
christine.gris@airbus.com

**Abstract.** Automation is widespread in interactive applications, promising multiple benefits to users, including enhancing comfort, safety, security and entertainment. Implicitly or explicitly, automation is now a critical design option for interactive application designers. Unfortunately, despite its long use (especially in safety-critical systems) assessing the benefits and the drawbacks of design alternatives including automation remains a craft activity, unsupported by conceptual frameworks or tools. In order to address this problem, we present the RCRAFT framework. The framework considers five attributes of automation: Resources, Control Transitions, Responsibility, Authority, and System Functions and User Tasks. We show how these attributes support the assessment of designs involving automation. Furthermore, adding the RCRAFT concepts to task models makes it possible to evaluate automation properties such as transparency, congruence and controllability in addition to usability. We demonstrate the utility of our approach in a case study, comparing the design for an existing Flight Warning System currently deployed in Airbus A350 against a redesigned version which incorporates functionality to provide recommendations to pilots handling unusual situations. We demonstrate that the RCRAFT framework helps in highlighting the implications of different design alternatives by making the impact of proposed changes on both users' work and the required properties of automated components explicit.

**Keywords:** Automation, aircraft cockpits, properties, tasks

## 1 Introduction

With different objectives in mind [31], interactive systems designers add autonomous behaviors in most of their designs. Recurring objectives are to increase user performance (e.g. automatic repetitive addition of a letter to a document when pressing a keyboard key for a long time), to increase comfort (e.g. adding a conveyor belt in an

airport) or to reduce errors (e.g. automatic detection of spelling errors in a word processing application). In the context of safety-critical systems, other objectives may be targeted, such as enhanced safety (e.g. the ABS Anti-lock Braking System in cars that prevents drivers from causing the wheels to lock by pressing the brake excessively) or increased security (e.g. auto-lock systems in most mobile phones to maintain the privacy and integrity of private data).

Such partly-autonomous behaviors may be added to any layer of the architecture of an interactive application [32]. For instance, automation can be added to the presentation part (e.g. automatically snapping a graphical object to a grid), to the dialogue part (e.g. automatic movement of the insertion point in the various text fields of a form) or to the functional core (e.g. autonomous disconnection from a server after too long a period of inactivity). Beyond these engineering considerations, changing automation designs might have a deep impact on operators' work and performance and not always in the expected way. Indeed, as demonstrated by Yerkes and Dodson [49] more automation might reduce operator vigilance (complacency) and prevent operators from reacting promptly to adverse events (loss of situation awareness). Finally, adding automation might also deeply alter the nature of the work itself. This was observed, for example, in the aviation domain [13] where pilots' work changed drastically from flying the aircraft (providing continuous input to the aircraft) to supervising systems of systems (monitoring information and reacting mostly to handle adverse events). Such changes also propagate to the way in which operators are selected, trained and deployed. Indeed, due to automation (and especially to the addition of Flight Management Systems) the standard flying crew in large civil airliners has been reduced from three to two.

These examples demonstrate that design decisions touching automation may have deep consequences, and that designers should clearly understand the scope of the options they are considering and identify their impact both on engineering aspects of the system and on the work context. HCI conceptual frameworks that support the design of interactive applications provide very limited guidance on automation design. For example, the ISO standard on human-centered design [23] explicitly mentions that it is important to define an appropriate allocation of functions and tasks between the system and the user, but does not provide guidance as to how this could be achieved. Another example is Action Theory [30], which supports the analysis of interactive systems from the perspective of users trying to reach their goals by perceiving the system's state and acting on it. This can help us to figure out possible misunderstandings when a user evaluates the system's state or errors when a user acts on the system, but it cannot help us compare two different automation designs in terms of their implications for the user's work.

To assess automation, user research (mainly via user testing) is generally performed in a similar way as for any kind of interactive system [**Erreur ! Source du renvoi introuvable.**], overlooking the specificities of automation even though some guidelines (focusing on AI systems and not automation) have been proposed to support heuristic evaluations by HCI experts [1].

This paper addresses that problem by providing means for evaluating the benefits and drawbacks of alternative automation designs. The work presented in this paper is part of a research project that aims to propose processes, techniques, methods and tools

to take automation into account while engineering interactive systems in the large civil aircraft domain. We propose the RCRAFT framework, which identifies five attributes of automation: Resources, Control Transitions, Responsibility, Authority and System Functions and User Tasks. We show how these attributes support the assessment of automation designs and more precisely how they clarify the effect of different design choices on users' activity. To this end, we extend a task-modelling notation to encompass each of the RCRAFT aspects, and demonstrate that these extended task models support the evaluation of automation-related properties such as transparency, congruence and controllability, as well as usability. These results allow us to answer the questions raised in the paper's title in a systematic way.

In the next section, we present the RCRAFT framework and outline the properties of interactive systems which are directly influenced by automation. We show how these concepts are necessary and sufficient to assess those properties. Section 3 presents the task-modelling notation extended with RCRAFT concepts. Section 4 introduces our avionics case study: the Flight Warning System used in large civil aircraft. We demonstrate the use of the RCRAFT framework on two variants of the Flight Warning System. Section 5 positions RCRAFT with respect to related work, while section 6 concludes the paper with a discussion of the contribution of this work.

## **2 The RCRAFT Conceptual Framework for Automation**

Whatever the design objectives are [31], the automation design will influence the underlying properties of the interactive systems. This section presents the RCRAFT conceptual framework, which provides a means to characterize the automation elements of interactive system designs. We demonstrate that the five concepts identified in the RCRAFT framework are sufficient to support the assessment of these properties. The last sub-section of this section presents a summary of the relationship between properties of interactive systems (influenced by automations) and the RCRAFT concepts.

### **2.1 Expected Properties for Automation**

Automation influences the behavior and the use of the entire interactive system but some properties are more influenced than others are. In this section, we list the main properties of semi-autonomous systems and identify the information required to assess them. Early work from IFIP Working Group 2.7/13.4 [17] introduced the notion of internal and external properties of interactive systems and [14] presented a notation to describe properties and how they are supported (or not supported) by various design options. Automation-related properties constitute a subset of these properties and focus "only" on some design aspects. For each of the selected properties, we propose a definition from the literature and make explicit the prominent concepts related to automation. We only consider here properties that can be assessed by objective measures and thus for which predictive assessment can be made. Indeed, subjective measures require the involvement of users/operators and rely on questionnaires and interviews techniques such as System Usability Scale [7] (to assess user satisfaction) and Attrackdiff [19] (to assess user experience).

**Usability.** Usability is the “extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use” [22]. Efficiency and effectiveness are two contributing factors to usability that can be measured objectively. Efficiency addresses performance and errors (as cost of recovering from errors) while effectiveness addresses the number of operator tasks that are supported by the interactive system. To analyze effectiveness and efficiency we need to identify:

- the goals and tasks that the user needs to perform and their behavior,
- the functions that are embedded in the interactive system and their behavior,
- the resources (information, knowledge and objects) required to perform the tasks, both on the interactive system side and on the user’s side.

**Transparency.** Westin et al. [46] define automation transparency as “the automation’s ability to afford **understanding and predictions about its behavior**. It is a measure of the automation’s openness in information **communicated, through the interface, to the operator**: *what* the automation is currently doing, *which information is being used, how it is being processed*, and *when it is provided*.” Predictability and comprehensibility also contribute to transparency. Indeed, McDermott et al. [28] define predictability as “the transparency of future intentions, states, and activities of the automation partner”.. Cramer et al. [10] define comprehensibility as the extent to which the user understands the meaning of the system feedback. In order to analyze transparency, we need to identify:

- the description of the system behavior and how it displays functions allocated to the system to the operator,
- the resources (information, data, objects) used by the user to perform their tasks and processed by the system,
- which information managed by the system is presented to the user, how it is presented and when,
- the detail of the execution of the functions and especially when functions are made available to the user,
- whether, and when, it is possible for the user and/or the system to interrupt one another, and how behavior is resumed after an interruption has occurred [3].

**Congruence.** Hollnagel [21] defines congruence as a matching between user tasks and system functions: “how the **capabilities of humans and machines should be matched** to achieve the operational system goals over a range of situations”. To analyze congruence we need to identify:

- the tasks that are performed by the users and the functions offered by the system,
- the information managed by both entities,
- the requirements of each entity for information and functionality from the other entity, and how assurance will be given that these requirements are satisfied.

**Controllability.** Roy et al [39] define the controllability of an automated system as “how much a **user is in control of the process**” and “to what extent an **automated result can be manually modified**”. To analyze controllability we need to identify:

- which manual and interactive tasks trigger the execution of functions in the system,

- whether or not each system function could be interrupted by some interaction from the user (defined as “full control” in [27]),
- the tasks or functions that have an impact on the resources manipulated by the system or the user.

**Accountability.** Several contributions (e.g. [42], [11]) address the issue of the accountability of automated systems, but they only consider it as a social construct influencing the way the operator performs their activities. Indeed, [42] shows that adding accountability to operators decreases their speed in decision making and might also decrease the quality of their decisions due to the “passing the buck” effect [43] (i.e. postponing decisions or trying to delegate accountability to others). According to the Oxford Dictionary, accountability is “the fact of being **responsible** for your decisions or actions and expected to **explain** them when you are asked”. To analyze accountability we need to identify:

- what is the expected result of the work performed (there may be more than one),
- which entity (system or user) is performing which action (function or task),
- which action from an entity influences the expected result of the work.

## 2.2 RCRAFT - Allocation of Functions and Tasks (FT)

The FT part of the RCRAFT conceptual framework deals with the issue of Allocation of Functions and Tasks. The generic term is usually function allocation [47] but RCRAFT differentiates the term function according to the entity that performs it. Activities performed by human operators are called “tasks” while the ones performed by the interactive system are called “functions”. Automation designs will thus result in the identification of which activities are carried out by which entity (user or system or both). RCRAFT-based design of automation is thus based on the identification and the complete description of the human-system cooperative work as tasks and functions.

## 2.3 RCRAFT - Allocation of Resources (R)

The “R” part of RCRAFT deals with the identification and representation of resources used by the entities. “Resource” is a generic term used to cover data, information (in the head of the user), physical objects (e.g. a credit card), software objects (information manipulated by the system) and devices (input, output and input/output) required to perform the system functions and the human tasks. As for the “FT” part, automation designs will allocate resources to entities, which may or may not share it with other entities. In [26], the authors have shown that, in order to describe operators’ tasks precisely, identification and representation of data related to those tasks are crucial.

## 2.4 RCRAFT - Allocation of Authority (A)

The “A” part of RCRAFT is concerned with identifying the controlling entity (i.e. which entity can influence the situation so that it develops or continues in a way which

satisfies its requirements [16]). The entity with authority performs or defines constraints on the tasks and functions that modify the state of a system, an object or the environment to reach a goal. These constraints, tasks or functions affect the resources needed by the user or the system. Automation designs may allocate authority globally (i.e. one entity is considered as master and the other as slave [50]). This allocation may be static or dynamic/adaptive [48] i.e. changing over time to adapt to changes (context, workload ...).

## 2.5 RCRAFT - Allocation of Control Transition (CT)

The “C” part of RCRAFT deals with the issue of Control Transition, which defines how an entity may takeover control, hand it over or share it with another entity [44]. Automation designs will identify Control Transitions, which describe who can modify the allocation of control. As discussed above, it is possible that the entity that performs a task or a function that initiates a control transition may not have authority. An entity may release control without defining how control is to be allocated going forward, leaving that task to another entity with authority. This justifies the conceptual separation of the two entities.

## 2.6 RCRAFT - Allocation of Responsibility(R)

The second “R” in RCRAFT deals with the issue of Responsibility, which defines which entity can derive a specific outcome on which the user goal depends. This outcome is called the ‘result’. Automation designs will identify the list of expected results for the work, and will indicate which activity of which entity influences one of the expected results of the work carried out jointly by the system and the user. The entity that influences one of the expected results will be said to be (at least partly) accountable for the outcome.

## 2.7 RCRAFT –Relationship to Automation Properties

**Table 1** summarizes the correspondence between the RCRAFT concepts and the properties of partly-autonomous interactive systems. It is important to note that all of the concepts of RCRAFT are needed to assess the properties identified in section 2.1 and that each property is influenced by at least two RCRAFT concepts.

**Table 1.** Table summarizing the relationship between RCRAFT concepts and the properties.

	Re-sources	Authority	Control Transitions	Responsibility	Functions and Tasks
Usability	X				X
Transparency	X	X	X	X	X
Congruence	X		X		X
Controllability		X	X		X
Accountability		X	X	X	



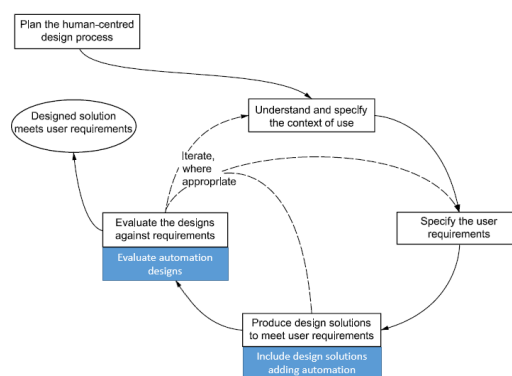
For transparency, all of the RCRAFT concepts are required, which demonstrates that designing interfaces for partly-autonomous systems necessitates consideration of multiple complex concepts. This fact explains why so much work on the assessment of automation has exhibited bad designs, even in largely distributed systems such as Microsoft Excel [51] and Microsoft Windows [3]. Further, it demonstrates that many of the notable classes of operator errors [33] identified in the context of automation [40] can be explained by this complexity, rather than solely by poorly done design activities.

This conceptual framework is based on the work from [2]. In addition to the study of relationships between properties and the main concepts of allocation of automation, the work presented in this paper refines the concept of authority by explicitly highlighting the allocation of control transition. The work presented in this paper also provides a more adapted technique for modelling the allocation of tasks and functions.

It is important to note that this set of properties has only been selected as they are very often put forward by researchers and practitioners in automation e.g. [21] and [3]. As mentioned in the introduction of this section, other properties, such as user experience [38] or trust [29], could have been added, but due to their subjective nature we have not included them in this paper. However, we believe that the constructs inside the RCRAFT framework can be positioned with respect to these properties too, as these concepts constitute the building blocks of automation designs.

## 2.8 RCRAFT – Relationship with UCD Process

**Fig. 1** indicates where RCRAFT could be used with ISO 9241-part 210 [23]. The blue boxes represent aspects added to the original User Centered Design process. That standard explicitly mentions function allocation in section 6.4.2.2., where it is treated as a sub part of the “Produce design solutions to meet requirements” phase (bottom of **Fig. 1**). However, the next phase “Evaluate the designs against requirements” does not mention how to evaluate automation designs and how these designs impact user requirements. This is where RCRAFT would be useful in partitioning automation into core concepts that can be designed and assessed independently.



**Fig. 1.** Positioning RCRAFT usage into UCD process (from [23])

### 3 Modelling and Analysis of Automation in an Interactive System

As identified in the previous section, the analysis of automation properties requires identification and description of user tasks, system functions, their refinement and their temporal ordering, as well as identification of the data and objects required to perform these tasks and functions. For that purpose, we need a declarative and procedural language, which has the capacity to describe interruptions (such as possible handovers or takeovers).

Task models aim to represent, in a hierarchical and temporally ordered way, the tasks that the user performs to reach a goal. Task models may contain several types of tasks such as user tasks, abstract tasks, interactive tasks and system tasks. Interactive tasks enable us to represent specific interactions with the system, such as an input (e.g. press a button) or an output (e.g. display an alarm). When needed to achieve the user's goal, task models may include pure system functions (called system tasks in CTT [35]) such as data or command processing (e.g. trigger the opening of landing gear). We call this type of task models "integrated task models".

This integrated modeling approach offers a blended view of all the user tasks, their supporting system functions and the temporal ordering between them. However, the models usually focus on describing user activity and leave system description at a higher level of abstraction.

#### 3.1 Segregated Models and Comparison Process

In order to analyze the automated aspects of the interactive system and, more specifically, the allocation of functions and tasks to each entity, we need to produce a more detailed representation of system functions than that usually provided by integrated task models. Refining system tasks within the same model would greatly increase the complexity of the models and make them difficult to modify and understand. For this reason, we propose to model the system tasks and the operator tasks in two different models that we call the segregated models of tasks and functions. In the same way as user task models are defined for each actor and for each role of that actor, functions models are decomposed into the various roles that the system may play.

This is in line with previous work from Barboni et al. [2], which proposed the use of a combination of a user task model and a system model to assess the consistency and compatibility of user tasks and interactive system behavior. Similarly, Campos et al. [9] have shown that such separation of concerns in different models supports predictive assessment of the effectiveness dimension of usability. However, these previous approaches used different notations for the descriptions of the user tasks and the system ones. Our approach proposes to use the same user tasks-based notation for both.


To compare the two systems, there are 3 activities to carry out:


- First, produce segregated user tasks and system functions for each of the actor roles. They explicitly contain all the resources, tasks and operators describing their temporal relationships. These models encompass the description of the communications between those segregated models.


- Second, integrate each RCRAFT component in the segregated models.
- Third, produce a comparison table of the RCRAFT concepts for both systems under investigation.


### 3.2 Notation for Resources, Authority, Control Transitions, and Responsibility

In order to support the implementation of RCRAFT concepts, a notation needs to be capable of describing both user tasks and system functions, as well as resources that are necessary to perform user tasks and system functions (e.g. information, knowledge, physical or software objects [26]). In addition, tool support needs to be available for the notation, in order to facilitate the description of large numbers of elements, sharing the models between different stakeholders, and amending or reusing models easily. We selected the tool-supported notation HAMSTERS-XL because it fulfills all of these needs. However, in principle, it would be possible to use and extend any other similar notation. We extended HAMSTERS-XL to support the editing of segregated tasks and function models. Furthermore, HAMSTERS-XL and its associated tool HAMSTERS-XLE supports the customization of task types and data types [25] which we exploited to customize task types and to add new resource types needed to model automation aspects (described below).

**Representation of resources.** HAMSTERS-XL allows us to represent data, objects and devices manipulated by the user and by the system [26]. Resources are represented by labels preceded by the abbreviation of the data type, such as an ‘information’:  **Inf** : Selected recovery action, which is data that can be needed or modified by the user. Arcs between tasks (or functions) and data indicates whether the data is needed to perform the task (or function) or is modified by the performance of the task (or function). Required data is indicated by an arrow pointing to the task (or function), and modified data by an arrow pointing to the data. Lines connecting input or output devices to tasks indicate that the devices are required to perform the task.

**Representation of authority.** The symbol  placed on the right-hand side of a task (respectively function) represents the fact that the user (or system) has the authority to perform this task (respectively function) and that this particular task (or function) will affect the resources needed by the user or by the system.

**Representation of control transitions.** The symbol  placed on the left-hand side of a task (or function) represents the fact that the user (or system) has the authority to take over on this task (or function).

**Representation of Responsibility.** The symbol  placed on the right-hand side of the task (or function) represents the fact that the user (or system) has the responsibility to perform this task (or function). Two additional elements of the notation - expected results and actual results (which are represented as resources) - allow us to describe the expected result when performing a task (or function). The expected result is

indicated by an arrow terminating at the expected result element. Modification of the actual results by a task (or function) is indicated by an arrow terminating at the result.

## 4 Application of the RCRAFT Framework and Process on the Warning System in Commercial Aircraft

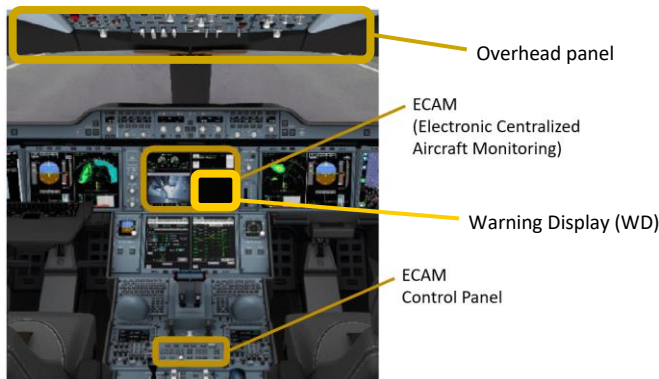
In current large civil aircraft, the flying crew is alerted to potential problems with aircraft systems (e.g. engines, air conditioning etc.) via a centralized monitoring and alert system called the Flight Warning System (FWS).

In this section, we present the current FWS of A-350 aircraft cockpit (A350-FWS) and a recommendation-based FWS (REC-FWS). We applied the RCRAFT framework and process to the analysis of the A350-FWS and the REC-FWS. For the purpose of the analysis, we consider each of these systems as an actor with the role of managing the alarms and guiding pilots in resolving them. On the user side, we detail the tasks of the “pilot monitoring” role (PM) who is in charge of managing the systems.

### 4.1 Principles of the Flight Warning System

The Flight Warning System automates some of the tasks previously allocated to the flight engineer, managing system failures by filtering and sorting the alarms triggered by faulty systems. This filtering and sorting process relies on:

- The priority level of the alarm: a **predefined absolute ranking (priority)** of alarms,
- **Inhibition** and **combination rules** (in case of the presence of other alarms),
- The **current flight phase**: some alarms are deferred so as to not disturb the crew (e.g. during take-off when full attention is required).



**Fig. 2.** ECAM in the A350 aircraft cockpit

In addition, the FWS displays different types of information: the procedure corresponding to the alarm, alarm titles and other relevant information about the current context (for example, limitations on the systems) on the Warning Display (WD) of the Electronic Centralized Aircraft Monitoring (ECAM, located in the center of the cockpit

as shown in **Fig. 2**). Finally, FWS triggers attention getters (see the Master Warning and Master Caution visual attention getters in **Fig. 4** items numbered 1 and 2).



**Fig. 3.** FWS user interface prototypes a) A350-FWS user interface prototype b) REC-FWS user interface prototype

## 4.2 The A350-FWS User Interface

**Fig. 3a)** presents a screen shot of the display of a procedure in the Airbus A350. The procedure contains a list of recovery actions to be performed by the pilot to handle an alarm. The procedure here is associated with the alarm “CAB PRESS EXCESS CAB ALT”, which states that there is a cabin air pressure problem due to the altitude. The first action tells pilots to use the oxygen masks (line CREW OXY MASKS USE). The “ENG ALL ENGS FLAME OUT” alarm is priority 20 and is raised when both engines are shut down or fail in flight, while the “CAB PRESS EXCESS CAB ALT” alarm is priority 5. If these two alarms are active at the same time, the FWS will display “CAB PRESS EXCESS CAB ALT” before “ENG ALL ENGS FLAME OUT”. In this case, only the procedure associated with the alarm “CAB PRESS EXCESS CAB ALT” will be displayed see **Fig. 3.a.**)

To access the procedure associated with the next alarm, the pilots must perform each action line of the displayed procedure and clear this procedure. Completion of a recovery action is either sensed by the FWS or has to be validated (the pilot presses the validation button on the ECAM Control Panel (ECP - located at the bottom of **Fig. 1**). The pilot can browse the active alarms with the scroll wheel on the ECP (see **Fig. 4** item 5). To clear a procedure and its associated alarm, the pilot pushes the CLEAR push button ECP (see **Fig. 4** item 3).

Pilots must double-check the active alarm on the ECAM with the overhead panel (top of **Fig. 2**). If the information is contradictory, the flight crew can declare the alarm to be spurious. The flight crew can discard a cancelable spurious alarm by pressing the EMER CANCEL push button of the ECP (see **Fig. 4** item 4).

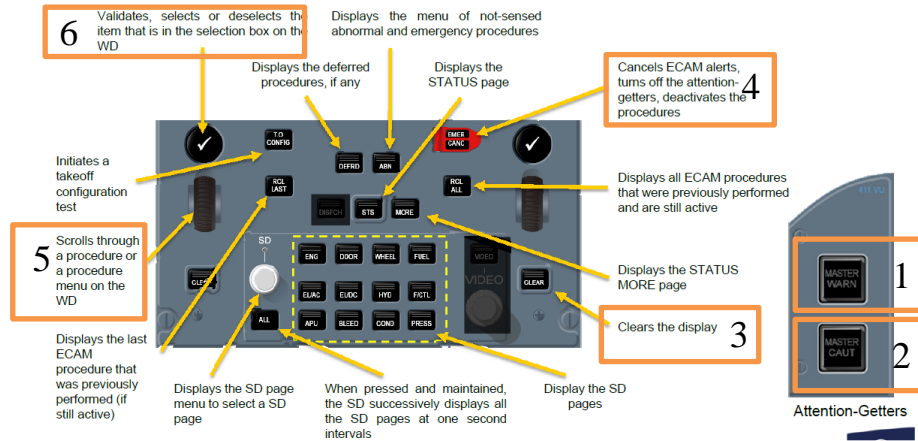


Fig. 4. ECAM Control Panel (ECP) description

### 4.3 A Recommendations-based FWS Prototype

In this section, we present a prototype (called REC-FWS) that differs from the A350-FWS previously presented. There are two main differences:

- Instead of presenting only the procedure associated with the highest priority alarm, the prototype offers a set of recommendations [37] from which the pilots will select.
- Instead of presenting recovery actions as full procedures, procedures are grouped into sets of meaningful recovery actions.

Thanks to this organization, pilots are able to select recovery actions based on the recommendations from the FWS but also on contextual information not available to the FWS, such as the health status of passengers. As in the A350-FWs, the REC-FWS filters the recovery actions. The filtering of recovery actions uses constraints similar to the filtering performed by the A350-FWS. The REC-FWS selects sets of recovery actions associated with the active alarms and sorts them according to a series of rules. There are two levels of criticality for the recommendations: immediate and not immediate. Each recommendation is allocated a level of criticality, which is used to group them on the screen (see Fig. 3.b). In Fig. 3.b, the REC-FWS proposes two recommendations for the FLY goal: “divert and glide” or “emergency descent”. Based on the knowledge of the flight crew about the proximity of the nearest airport, the flight crew may choose one or the other.

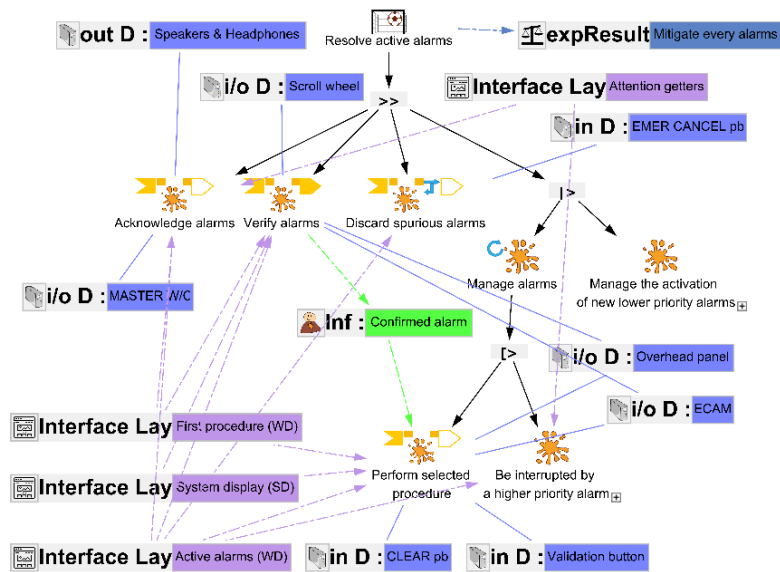
The ECP scroll wheel allows pilots to select a recommendation and validate when a recovery action has been performed. When all of the actions have been performed, the pilots clear the recommendation using the CLEAR button. The EMER CANCEL button allows pilots to discard spurious recommendations.

### 4.4 Segregated models of the pilot monitoring and A350-FWS

Fig. 5 presents the segregated tasks model describing the tasks allocated to the pilot monitoring (PM) in order to resolve alarms. The main goal “Resolve active alarms” is

decomposed into a sequence (“>>” operator) of several subroutines. A subroutine is a task that points out to another task model, in order to support the structuring and reuse of models [31].

The main goal is connected to the expected result “Mitigate every alarm” because it is an expected result when performing tasks to reach this goal. The first subroutine “Acknowledge alarms” describes the tasks to perceive, understand and silence the alarms with the Master W/C push button. This is captured in **Fig. 5** by the arc between this subroutine and the input/output device “MASTER W/C” and the interface layer “Attention getters”. The second subroutine describes the tasks of the PM to verify the alarms. The PM checks that the alarm is not spurious. S/he compares the system display (SD) with the system states displayed on the overhead panel. Third, the PM can discard a spurious alarm by pressing the EMER CANCEL push button. Then, to manage all the active alarms the PM performs the procedure selected by the FWS. This subroutine is described in **Erreur ! Source du renvoi introuvable.** When the PM has finished all the recovery actions in the procedure, s/he clears it to be able to perform the next one. This subroutine can be interrupted by a higher priority alarm (“[>” operator). If the priority of the incoming alarm is lower than the alarm associated with the current procedure, the PM can resume her task (“[>” operator).



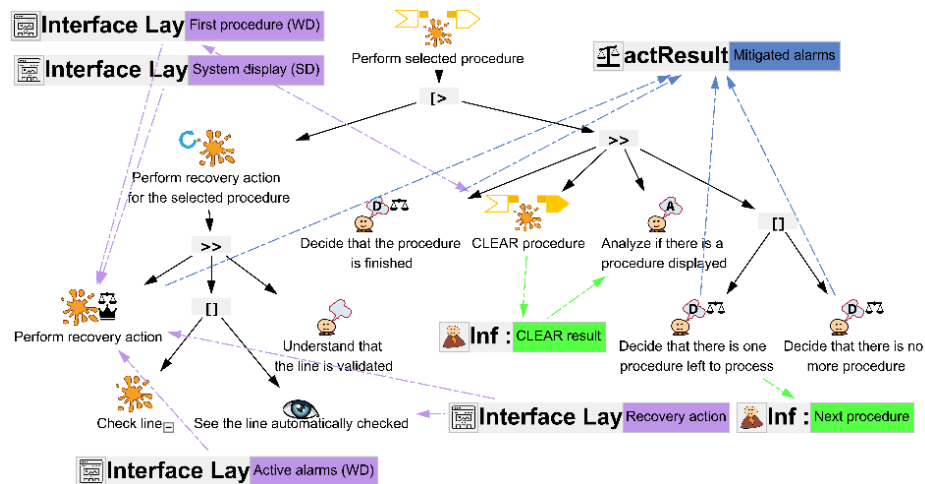
**Fig. 5.** PM-Resolve active alarms with A350-FWS: segregated tasks model of the “pilot monitoring” role to resolve active alarms with the former warning system

The following resources are accessible to the PM:

- The system states (through the interface layer: “System Display (SD)”, output device “ECAM” and input/output device “Overhead panel”),
- The active alarms (through the interface layer: “Active Alarms (WD)” and “Attention getters” and output device “Master W/C”, “ECAM” and “Speakers & Headphones”)

- The procedure (through the interface layer: “First procedure (WD)” and output device “ECAM”): selected set of recovery actions.

**Fig. 6** presents the subroutine “Perform selected procedure”. The PM has the authority to perform the recovery actions (the “Perform recovery action” abstract task). The PM does not have the authority to define the recovery action (i.e. content and execution order) but the PM performs it manually. The PM has the responsibility to perform the recovery actions and to decide whether there are still procedures to perform. If the PM makes an error on these tasks, it will have an impact on the actual result (mitigated alarms). For example, if the PM forgets to perform a procedure, an alarm will remain active.



**Fig. 6.** Perform selected procedure: subroutine of the “pilot monitoring” role to perform the selected procedure

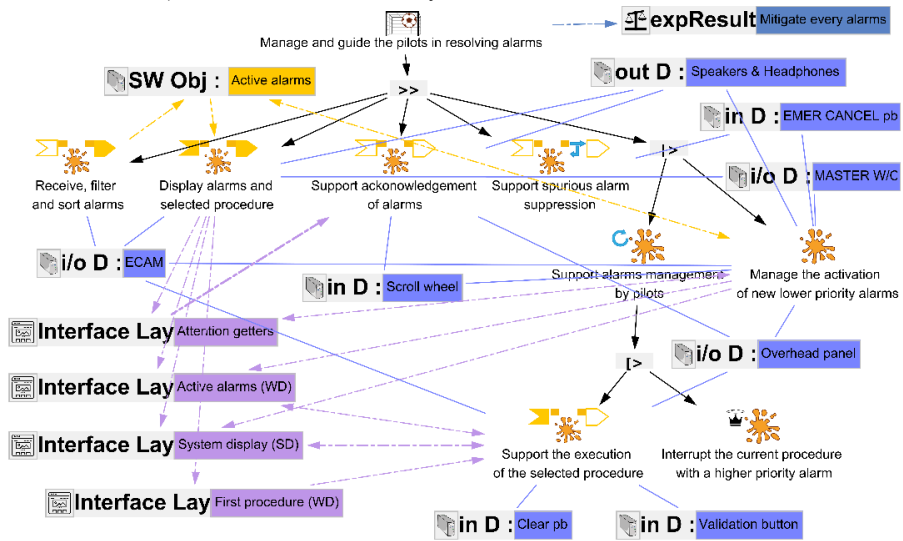
**Fig. 7** presents the functions model describing the functions allocated to the A350-FWS. First, the A350-FWS receives, filters and sorts the alarms as described in section 4.2. Second, the A350-FWS displays alarms and the selected procedure. This subroutine is described in **Fig. 8**. Third, the A350-FWS supports the acknowledgement of the alarms by cutting the attention getters. Fourth, the A350-FWS supports the suppression of spurious alarms if necessary. Then, the FWS-A350 supports the execution of the procedure by the PM until it receives a new alarm. If the new alarm has a higher priority than the alarm associated the current procedure, the A350-FWS initiates a control transition, which interrupts the support to the operator for the execution of the procedure.

The A350-FWS shares resources with the PM:

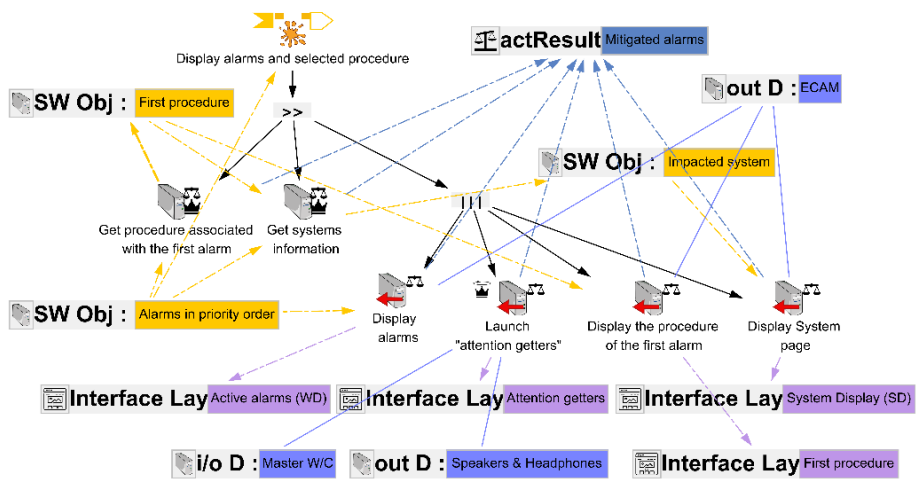
- The system states (through the interface layer: “System Display (SD)”, output device “ECAM” and input/output device “Overhead panel”)
- The active alarms (through the interface layer: “Active Alarms (WD)” and “Attention getters” and output device “Master W/C”, “ECAM” and “Speakers & Headphones”)



- The procedure (through the interface layer: “First procedure (WD)” and output device “ECAM”): selected set of recovery actions



**Fig. 7.** A350-FWS-Manage alarms and guide the pilots in resolving alarms: the model of functions of the A350-FWS role to manage alarms and guide the PM in resolving alarms



**Fig. 8.** A350-FWS-Display alarms and selected procedure: subroutine of the A350-FWS role to display alarms and selected procedure

The A350-FWS has the authority and the responsibility to get the procedure associated with the first alarm and the system information to display. The A350-FWS defines a set of recovery actions to perform and the order of execution of these recovery actions through a procedure. The A350-FWS adds another constraint on the alarm resolution

task by masking the other procedures associated with alarms of lower priorities (it displays only the procedure associated with the higher priority alarm). The A350-FWS has the responsibility to display alarms, attention getters, the procedure associated with the first alarm and the impacted system page. The A350-FWS initiates a control transition when it launches the attention getters. After this control transition, only the pilots have the authority to cut the attention getters. The A350-FWS does not initiate a control transition when it displays the procedure of the first alarm. The PM does not have the authority to select another procedure for the tasks considered in our case study.

#### 4.5 Segregated models of the pilot monitoring and REC-FWS

Fig. 9 presents the segregated tasks model describing the tasks allocated to the PM to resolve alarms with the REC-FWS. The main goal is connected to the expected result “Mitigate every alarm” because it is an expected result when performing tasks to reach this goal. The first three subroutines of the main goal “Resolve active alarms” are the same as those with the previous system.

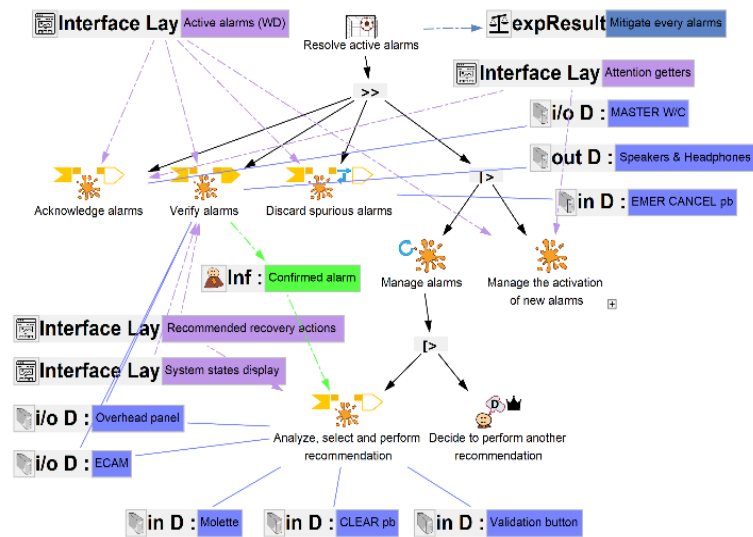
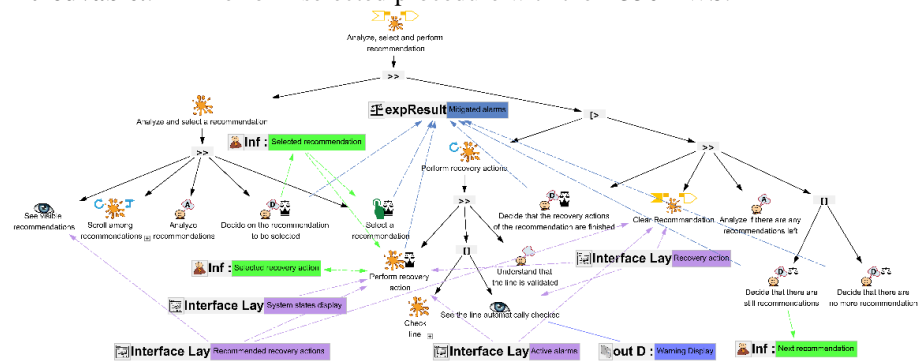


Fig. 9. PM-Resolve active alarms with REC-FWS: segregated tasks model of the “pilot monitoring” role to resolve alarms with the REC-FWS

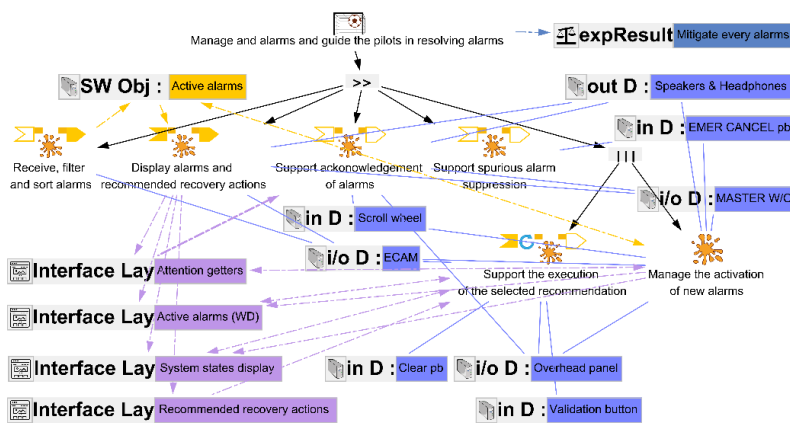
Then, the PM analyzes the options and selects and performs the recommended action to manage the alarm. This subroutine is presented in Fig. 11. The PM can interrupt himself (“[>” operator) whenever s/he wants. S/he has the authority to decide whether to perform another recommendation. The PM suspends the “manage alarms” iterative task when s/he needs to manage the activation of a new alarm. Once the new alarms have been acknowledged and verified, s/he can resume her task. The following resources are accessible to the PM:

- The system states (through the interface layer: “System states display”, output device “ECAM” and input/output device “Overhead panel”),
- The active alarms (through the interface layer: “Active Alarms (WD)” and “Attention getters” and output device “Master W/C”, “ECAM” and “Speakers & Head-phones”)
- The recovery actions (through the interface layer: “recommended recovery actions” and output device “ECAM”): all recovery actions associated with active alarms.

**Fig. 10** presents the PM’s subroutine “analyze, select and perform recommendation”. First, the PM analyzes and selects a recommendation. S/he perceives the recommended recovery actions through the interface layer “Recommended recovery actions”. Then, the PM has the authority and the responsibility to decide on the recommendation to be selected and to select this recommendation. The PM defines which recovery actions to perform and the order of execution of these sets of recovery actions. An error on these tasks can cause a derivation of the expected result “Mitigate every alarm”. The next tasks of this subroutine are the same as those presented in **Erreur ! Source du renvoi introuvable.**: PM-Perform selected procedure with the A350-FWS.



**Fig. 10.** PM-Analyze, select and perform recommendation: subroutine of the “pilot monitoring” role to choose, select and perform a recommendation



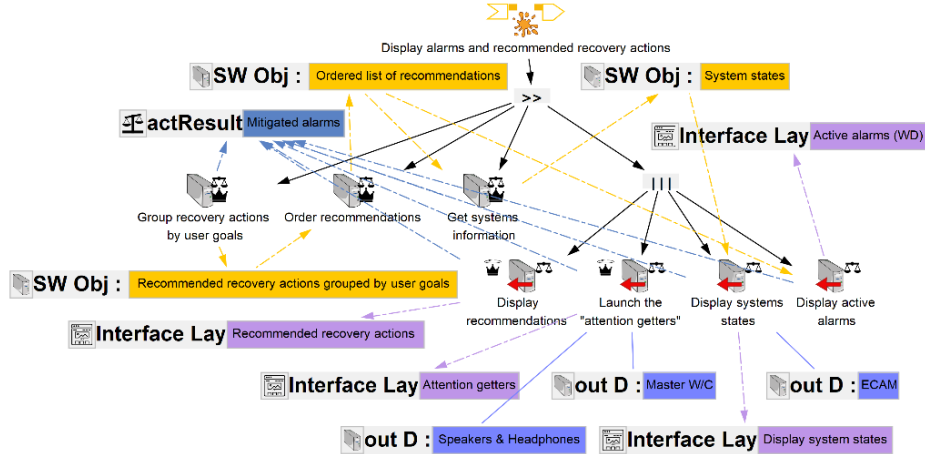
**Fig. 11.** REC-FWS-Manage alarms and guide the pilots in resolving alarms: model of functions of the REC-FWS role to manage alarms and guide the pilots in resolving alarms

**Fig. 11** presents the functions model describing the functions allocated to the REC-FWS. Contrary to the A350-FWS functions, the REC-FWS displays alarms and recommended recovery actions. In addition, the REC-FWS manages the activation of new alarms and supports the execution of the selected recommendation concurrently (“|||” operator).

The REC-FWS shares resources with the PM:

- The system states (through the interface layer: “System states display”, output device “ECAM” and input/output device “Overhead panel”),
- The active alarms (through the interface layer: “Active Alarms (WD)” and “Attention getters” and output device “Master W/C”, “ECAM” and “Speakers & Headphones”)
- The recovery actions (through the interface layer: “recommended recovery actions” and output device “ECAM”): all recovery actions associated with active alarms.

**Fig. 12** presents the subroutine describing the functions allocated to the A350-FWS to display alarms and recommended recovery actions. The REC-FWS has the authority and the responsibility to group recovery actions by user goals and to order recommendations. The REC-FWS defines sets of recovery actions and the execution order of the recovery actions within these sets (i.e. the recommendations). The REC-FWS defines an order of presentation for the recommendations. The REC-FWS initiates a control transition when it displays recommendations. After this control transition, the PM has the authority to select a recommendation and to define a different execution order from the one propose by the REC-FWS. The other functions of the REC-FWS are the same as those presented in **Fig. 8**: A350-FWS-Displays alarms and selected procedure.



**Fig. 12.** Subroutine of the REC-FWS role to display recommended recovery actions

#### 4.6 Comparison of both versions of the Warning Systems using RCRAFT

**Comparison of the allocation of functions and tasks.** The high-level hierarchy of allocation of PM tasks does not change. In the same way, the high-level hierarchy of allocation of functions to the flight warning system does not change between the A350-

FWS and REC-FWS versions. The main modification in task/function allocation concerns the tasks that deal with the actions performed to manage alarms: subroutine “Perform selected procedure” (in **Fig. 5** and **Erreur ! Source du renvoi introuvable.**) and subroutine “Analyze, select and perform recommendation” (in **Fig. 9** and **Fig. 10**). Specifically, the main modification concerns the functions that deal with supporting the execution of the selected procedure: subroutine “Display alarms and selected procedure” (in **Fig. 7** and **Fig. 8**) and subroutine “Display alarms and selected recommendations” (in **Erreur ! Source du renvoi introuvable.** and **Fig. 12**). For the REC-FWS, the number of PM tasks increases, with mainly cognitive tasks, as the PM has to analyze and select actions to perform to manage the alarms (**Fig. 10**). In particular, the PM has to browse recommendations as well as analyzing each of them, and to decide which to select. This is not the case with A350-FWS which proposes a procedure, a set of actions to perform to deal with an alarm, to the PM. The changes in function allocation between the A350-FWS and the REC-FWS concerns the system functions to prepare recommended actions (“Group recovery actions by user goals”, “Order recommendations”, in **Fig. 12**). For both versions of the system, the PM tasks match the system tasks, i.e. each system input function matches a user motoric task, and each user perceptive task matches a system output function. For example, the “Display recommendations” system output function in **Fig. 12** matches the “See visible recommendations” perceptive task in **Fig. 10**. The congruence properties are then met for each version of the system. The predictive cognitive load of the user is higher with the REC-FWS, which may have a negative effect on usability.

**Comparison of the allocation of resources.** The allocation of interactive devices does not change between A350-FWS and REC-FWS (e.g. devices “ECAM”, “Overhead panel”... in **Fig. 5** and **Fig. 9**) In both versions, the information about active alarms is shared between the PM and the system. The main change concerns the shared information. The REC-FWS version shares more information because it displays several possible actions to handle the alarms (information “Selected recommendation” and “Selected action” in **Fig. 10**), whereas the A350-FWS displays a predefined set of actions in a procedure (information “Next procedure” in **Erreur ! Source du renvoi introuvable.**) to target the solving of a specific alarm. The REC-FWS version is then more transparent than the A350-FWS.

**Comparison of the allocation of authority.** The allocation of authority moves towards the PM with the REC-FWS because more PM tasks have an effect on the result of the execution. The PM decides in which order to apply the actions to manage alarms (cognitive task “Decide on the recommendation to be selected” in **Fig. 10**), whereas it is not the case with the A350-FWS, where the procedure contains an ordered group of actions to perform. The REC-FWS version better fulfills the controllability property.

**Comparison of the allocation of control transitions.** The A350-FWS can interrupt the PM and take over by displaying a new procedure (task “interrupt the current procedure with a higher priority alarm” in **Fig. 7**) while the PM is handling a procedure, if a higher priority alarm is incoming. The REC-FWS hand over to the PM and releases control when it recommends actions to the PM (there is no interruption from the system

on the task “analysis, select and perform recommendation” in **Fig. 9**). Controllability is higher with REC-FWS than with A350-FWS.

**Comparison of the allocation of responsibility.** The expected results of the PM tasks and system functions are to mitigate every alarm (expected result “Mitigate every alarm in **Fig. 5** and in **Fig. 9**). The A350-FWS functions have an impact on the actual mitigation of the alarms (actual result “mitigated alarms” modified by functions in the subroutine “Display alarms and selected procedures” in **Fig. 8**). The system will then be accountable in case of any mitigation issues. Whereas it is the contrary with REC-FWS: in this case, the PM tasks have an impact on the actual result (actual result “mitigated alarms” modified by tasks in the subroutine “Analyze, select and perform recommendations” in **Fig. 10**).

## 5 Related Work addressing RCRAFT

Boy [Erreur ! Source du renvoi introuvable.] proposed a method to identify functions that could be performed by the system and tasks that could be performed by the user. Dearden et al. [12] refined the way in which the allocation of tasks is determined and by refining the types of functions and tasks. They proposed the IDA-S framework to support engineers in making early decisions about the requirements for automation. It identifies four types of tasks and functions (Information, Decision, Action and Supervision), allowing for finely-tuned allocations and a clearer understanding of how the allocations relate to taxonomies of levels of automation such as Parasuraman et al. [34]. This IDA-S framework has also been associated with UML notations to facilitate the description of allocations [18]. Beyond the analysis of task/function allocation, Boy [6] proposed a conceptual model to support the analysis of how authority is shared between humans and systems. This model comprises concepts that can support the understanding of allocation of authority when comparing several automation designs, however it does not provide explicit guidance for the analysis, and focuses only on authority. Heer [20] proposed recommendations for integrating AI based automation in interactive systems, which address many of the concepts we consider in RCRAFT. They do not provide explicit support for the analysis of the allocation in design solutions, as RCRAFT does. Pritchett et al [36] proposed the WMC (Work Model that Computes) simulation framework for analyzing the allocation of functions, authority and responsibility. This framework explicitly aims to take functions, tasks, authority and responsibility into account together. However, it is a computational approach to the analysis of automation. Tasks and functions are described programmatically so that a simulator produces the possible sequences of tasks and functions. The analysis has to be done “manually” using these sequences, with no explicit support for the analysis of the concepts of authority and responsibility. These existing approaches demonstrate that there is a need to take the RCRAFT properties into account. The approach presented in this paper takes them into account and in an integrated way.

Beyond the analysis of automation design, Wehrmeister et al. [45] proposed a model-driven approach for the implementation of automation systems. This approach

aims to address non-functional requirements about automation and to generate the embedded software to carry out the automated functions. However, it does not explicitly consider the RCRAFT concepts.

## 6 Conclusion and Perspectives

While designing automated functions is known to be a complex and error prone activity, no methods or tools are provided to support this task. The Human Factors community has provided high-level concepts to help understand automation and its impact on the work of operators, such as automation levels [34], metaphors [15] or frameworks [8]. More recent work has focused on the impact of automation designs on workload and as a consequence on its potential for reducing crew size [41].

This paper has presented a complementary, more pragmatic contribution to the assessment of automation designs. First, we proposed a decomposition of automation into five complementary concepts that must be considered while designing automations. We then showed how these concepts can be integrated into a task-modelling notation to represent both system and user behaviors and how these models can be analyzed to understand automation. We demonstrated the usefulness of the approach on a real case study in the domain of civil aviation by comparing the current A350 Flight Warning System with a revised prototype design, which exploits the principles of action recommendation outlined in the early parts of the paper. Using the approach, we have demonstrated that the FWS with recommendations gives more authority and more responsibility to the pilot than the current Airbus A350 FWS does on some specific tasks. In addition, the FWS with recommendations never takes over control from the pilot. Finally, the FWS with recommendations has a higher transparency (less filtering than the A350 FWS). It is important to note that higher transparency does not mean higher usability, as more information presented requires more scanning time and this might reduce task performance.

Even though the paper presents the application of the RCRAFT framework on a case study in the avionics domain, the framework integrates generic automation concepts and is thus applicable to other domains. For instance these concepts have recently been applied to analyze the differences between SAE J3016 levels of automation in the automotive application domain [5] and the levels of automation coming from the air traffic management domain [34].

This work is part of a more ambitious work in the area of command and control systems, targeting at providing notation and tools to support systematic approaches to the design and evaluation of automation, as well as to support the development and implementation of automation-rich interactive systems in their operational contexts [4].

**Acknowledgement.** The authors are deeply indebted with the shepherd of INTERACT 2021 program committee who suggested and made significant improvements in the preparation of the final version of this paper.

## References

1. Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, Paper 3, 1–13.
2. Barboni, B., Ladry, J-F, Navarre, D., Palanque, P., Winckler M.: Beyond modelling: an integrated environment supporting co-execution of tasks and systems models. ACM SIGCHI symposium on Engineering interactive computing systems (EICS '10). ACM, 165–174. (2010).
3. Bernhaupt, R., Cronel, M., Manciet, F., Martinie, C., Palanque, P.: Transparent Automation for Assessing and Designing better Interactions between Operators and Partly-Autonomous Interactive Systems. *5th International Conference on Application and Theory of Automation in Command and Control Systems (ATACCS '15)*. ACM, 129–139 (2015).
4. Bouzekri, E., Canny, A., Fayollas, C., Martinie, C., Palanque, P., Barboni, E., Deleris, Y., Gris, C. Engineering issues related to the development of a recommender system in a critical context: Application to interactive cockpits, *International Journal of Human-Computer Studies*, Volume 121, 2019, Pages 122-141.
5. Bouzekri, E., Martinie, C., Palanque, P. A-RCRAFT Framework for Analysing Automation: Application to SAE J3016 Levels of Driving Automation. Cristina Olaverri-Monreal, Fernando García-Fernández, Rosaldo J. F. Rossetti. *Human Factors in Intelligent Vehicles*, River Publishers, 2020, 9788770222037.
6. Boy, G.A.: Orchestrating Situation Awareness and Authority in Complex Socio-technical Systems. In: Aiguier, M. et al. (eds.) *Complex Systems Design & Management*. pp. 285–296 Springer, Berlin, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-34404-6\\_19](https://doi.org/10.1007/978-3-642-34404-6_19).
7. Brooke, J. System usability scale (SUS): a quick-and-dirty method of system evaluation user information. Reading, UK: Digital Equipment Co Ltd 43 (1986).
8. Bye, A., Hollnagel E., Brendeford, T.S.: Human-machine function allocation: a functional modelling approach, *Reliability Engineering & System Safety*, Volume 64, Issue 2, Pages 291-300, (1999).
9. Campos, J.C., Fayollas, C., Martinie, C., Navarre, D., Palanque, P., Pinto, M.: Systematic automation of scenario-based testing of user interfaces. 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '16). ACM, 138–148. (2016).
10. Cramer, S., Kaup, I., and Siedersberger, K.: Comprehensibility and Perceptibility of Vehicle Pitch Motions as Feedback for the Driver During Partially Automated Driving. *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 1, pp. 3-13, (2019).
11. Cummings, M. L. Automation and Accountability in Decision Support System Interface Design, *Journal of Technology Studies*, Vol. 32, Number 1, (2006).
12. Dearden, A., Harrison, M. D., Wright, P.C. Allocation of function: scenarios, context and the economics of effort. *Int. J. Hum.-Comput. Stud.* 52(2): 289–318 (2000).
13. Drogoul, F., Palanque, P.: How to make automation a good solution to the current problems in ATM? Hermes Air Transportation organization, April R19-PP/05, 7p. [http://hermes.aero/wp-content/uploads/2019/06/R19-PP\\_05-EUROCONTROL.pdf](http://hermes.aero/wp-content/uploads/2019/06/R19-PP_05-EUROCONTROL.pdf) (2019).
14. Fayollas, C., Martinie, C., Palanque, P., Aït-Ameur, Y. QBP Notation for Explicit Representation of Properties, Their Refinement and Their Potential Conflicts: Application to Interactive Systems. *16th IFIP Conference on Human-Computer Interaction (INTERACT)*, pp. 91-105, (10.1007/978-3-319-92081-8\_9) (2017).



15. Flemisch, F., Adams, C.A., Conway, S.R., Goodrich, K.H., Palmer, M.T., & Schutte, P.C.: The H-Metaphor as a Guideline for Vehicle Automation and Interaction. NASA Technical Report <https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20040031835.pdf> (2005)
16. Flemisch, F., Heesen, M., Hesse, T., Kelsch, J., Schieben, A., Beller, J.: Towards a dynamic balance between humans and automation: authority, ability, responsibility and control in shared and cooperative control situations. *Cogn Tech Work.* 14, 3–18. <https://doi.org/10.1007/s10111-011-0191-6> (2012).
17. Gram, C., Cockton, G.: Internal Properties: The Software Developer's Perspective. In *Design Principles for Interactive Software* (eds), pp. 53-89, Springer US (1996).
18. Harrison, M.D., Johnson, P.D., Wright, P.C. Relating the automation of functions in multi-agent control systems to a system engineering representation. *Handbook of cognitive task design.* 503–524 (2003).
19. Hassenzahl, M.: The effect of perceived hedonic quality on product appealingness. *International Journal of Human-Computer Interaction*, 13, 481–499 (2001).
20. Heer, J.: Agency plus automation: Designing artificial intelligence into interactive systems. *PNAS.* 116, 6, 1844–1850 (2019). <https://doi.org/10.1073/pnas.1807184115>.
21. Hollnagel, E.: From function allocation to function congruence. In: S. Dekker & E. Hollnagel, Eds. *Coping with Computers in the Cockpit.* Aldershot, U.K. Ashgate (1999).
22. International Organization for Standardization. *Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts, ISO 9241-11:2018(E)*, ISO, (2018).
23. ISO. "ISO 9241-210:2019". ISO. International Organization for Standardization. Retrieved 17 February 2020. <https://www.iso.org/standard/77520.html>
24. Martinie, C., Palanque, P., Barboni, E. and Ragosta, M, Task-model based assessment of automation levels: Application to space ground segments, 2011 IEEE International Conference on Systems, Man, and Cybernetics, 2011, pp. 3267-3273.
25. Martinie, C., Palanque, P., Bouzekri, E., Cockburn, A., Canny, A., Barboni, E.: Analysing and Demonstrating Tool-Supported Customizable Task Notations. *PACM on Human-Computer Interaction*, Vol. 3, EICS, Article 12, 26 pages (2019).
26. Martinie, C., Palanque, P., Ragosta, M., and Fahssi, R.: Extending procedural task models by systematic explicit integration of objects, knowledge and information. 31st European Conference on Cognitive Ergonomics (ECCE '13). ACM, Article 23, 1–10 (2013).
27. Maudoux, G., Pecheur, C., Combéfis, S. Learning Safe Interactions and Full-Control. *Handbook of Formal Methods in Human-Computer Interaction*, 297-317 (2017).
28. McDermott, P., Dominguez, C., Kasdaglis, N., Ryan, M., Trhan, I., Nelson, A.: *Human Machine Teaming Systems Engineering Guide*, MP180941, The MITRE Corporation, McLean, VA (2018).
29. Mirmig, A., Gärtner, M., Laminger, A., Meschtscherjakov, A., Trösterer, S., Tscheligi, M., McCall, R., and McGee, F.: Control Transition Interfaces in Semiautonomous Vehicles: A Categorization Framework and Literature Analysis. 9th Int. Conf. on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI '17). ACM, 209–220 (2017).
30. Norman, D. A., *The Design of Everyday Things.* New York: Basic Book, 1988.
31. Palanque P.: Ten Objectives and Ten Rules for Designing Automations in Interaction Techniques, User Interfaces and Interactive Systems. In *Proceedings of the International Conference on Advanced Visual Interfaces (AVI '20)*. ACM, Article 2, 1–10 (2020).
32. Palanque, P.: Engineering Automations: From a Human Factor Perspective to Design, Implementation and Validation Challenges. ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '18). ACM, Article 2, 1–2 (2018).
33. Palmer, E.: Oops, it didn't arm'- A case study of two automation surprises. In *International Symposium on Aviation Psychology, 8th*, Columbus, OH (pp. 227-232) (1995).

34. Parasuraman, R., Sheridan, T.B., Wickens C.D.: A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans* 30(3):286–297 (2000).
35. Paternò, F., Mancini, C. & Meniconi, S. ConcurTaskTree: a Diagrammatic Notation for Specifying Task Models. *IFIP TC 13 Int. Conf. on Human-Computer Interaction (INTERACT 1997)*. Chapman & Hall, 362–369 (1997).
36. Pritchett, A.R., Kim, S.Y., Feigh, K. Modeling Human–Automation Function Allocation. *Journal of Cognitive Engineering and Decision Making*. 8, 1, 33–51 (2014). <https://doi.org/10.1177/1555343413490944>.
37. Ricci, F., Rokach, L., & Shapira, B.: Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook* (p. 1-35). Springer, Boston, MA. (2011).
38. Roto, V., Palanque, P., Karvonen, H.: Engaging Automation at Work – A Literature Review. In: Barricelli, B.R. et al. (eds.) *Human Work Interaction Design. Designing Engaging Automation*. pp. 158–172 Springer International Publishing, Cham (2019). [https://doi.org/10.1007/978-3-030-05297-3\\_11](https://doi.org/10.1007/978-3-030-05297-3_11).
39. Roy, Q., Zhang, F., Vogel, D.: Automation Accuracy Is Good, but High Controllability May Be Better. In *Proc. of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, Paper 520, 1–8 (2019).
40. Sarter, N., Woods, D., Billings, C. E.: Automation surprises. *Handbook of human factors and ergonomics*, 2, 1926-1943 (1997).
41. Schmid, D., Korn, B., Stanton, N.A.: Evaluating the reduced flight deck crew concept using cognitive work analysis and social network analysis: comparing normal and data-link outage scenarios. *Cogn Tech Work* **22**, 109–124 (2020).
42. Skitka, L. J., Mosier, K., Burdick, M.D.: Accountability and automation bias, *International Journal of Human-Computer Studies*, Volume 52, Issue 4, pp 701-717 (2000).
43. Steffel M., Williams E. F., Permann-Graham J. Passing the buck: Delegating choices to others to avoid responsibility and blame, *Organizational Behavior and Human Decision Processes*, Volume 135, 2016, Pages 32-44, <https://doi.org/10.1016/j.obhdp.2016.04.006>.
44. Tan, D., Chen, W., Wang, H., Gao, Z.: Shared control for lane departure prevention based on the safe envelope of steering wheel angle, *Control Engineering Practice*, Volume 64, Pages 15-26 (2017).
45. Wehrmeister, M.A., Pereira, C. E., Rammig, F. J. Aspect-Oriented Model-Driven Engineering for Embedded Systems Applied to Automation Systems. *IEEE Transactions on Industrial Informatics*. 9, 4, 2373–2386 (2013). <https://doi.org/10.1109/TII.2013.2240308>.
46. Westin, C., Borst, C., Hilburn, B.: Automation Transparency and Personalized Decision Support: Air Traffic Controller Interaction with a Resolution Advisory System, *IFAC-PapersOnLine*, Volume 49, Issue 19, 2016, Pages 201-206 (2016).
47. Wright, P., Fields, R., Harrison, M. Analyzing Human-Computer Interaction as Distributed Cognition: The Resources Model. *Hum. Comput. Interact.* 15(1): 1-41 (2000).
48. Wu, Y., Wei, H., Chen, X., Xu, J., Rahul, S.: Adaptive Authority Allocation of Human-Automation Shared Control for Autonomous Vehicle. *Int.J Automot. Technol.* 21, 541–553 (2020).
49. Yerkes, R.M., Dodson, J.D.: "The relation of strength of stimulus to rapidity of habit-formation". *Journal of Comparative Neurology and Psychology* 18: 459–482 (1908).
50. Zhang, Z., Zhao, D.: Master-slave control strategy of tele-manipulator. *International conference on Robotics and biomimetics (ROBIO'09)*. IEEE Press, 2063–2067 (2009).
51. Ziemann, M., Eren, Y. & El-Osta, A. Gene name errors are widespread in the scientific literature. *Genome Biol* 17, 177 (2016). <https://doi.org/10.1186/s13059-016-1044-7>.