# Using Task Descriptions with Explicit Representation of Allocation of Functions, Authority and Responsibility to Design and Assess Automation

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

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

**Abstract.** Automation can be considered as a design alternative that brings the benefits of reducing the potential for human error and of increasing performance. However, badly designed automations, of which some of them are called automation surprises, can have a very negative impact on the overall performance of the couple operator/system. Automation design requires the definition of three specific aspects defining the relationship between the user and the system: allocation of functions, authority and responsibility. While these abstract concepts are usually well understood at a high level of abstraction, their integration within a development process is cumbersome. This paper presents an approach based on task models to explicitly handle those concepts. We show how such concepts can be integrated in a task modeling notation and illustrate on a case study how this notation can be used to describe design alternatives with different allocation of functions, authority and responsibility between the user and the system. Exploiting the case study, we demonstrate that embedding explicitly these concepts in a notation supports analysis and assessment of automation designs.

**Keywords:** Automation design and assessment · Task modeling
Allocation of functions · Authority · Responsibility

## 1 Introduction

Currently, automation is one of the main means for supporting operators using systems that feature increasing complexity. Automation makes it possible for designers to transfer the burden from operators to a system by allocating to the system tasks that were previously performed by the operator. Automation is defined as “the technique, method or system of operating or controlling process by highly automatic means, as by electronic devices, reducing human intervention to a minimum” [9]. In this definition, the concept of

© IFIP International Federation for Information Processing 2019
Published by Springer Nature Switzerland AG 2019
B. R. Barricelli et al. (Eds.): HWID 2018, IFIP AICT 544, pp. 36–56, 2019.
https://doi.org/10.1007/978-3-030-05297-3_3

control is highlighted in addition to the concept of allocating functions to the system. This concept of control is related to the authority the human or the system may have on the triggering of an operation or of a process. Another term used to define automated systems is “autonomy”. This term refers to the independence from outside control of the human or system entity (i.e. self-directedness), whereas automation refers to an entity that will do only what it is programmed to do without having any choice [25]. This implies that the automated system may have a certain level of independence and thus be responsible for the outcome of the execution of the triggered functions. As both the operator and the automated system may have authority for triggering functions, then both the operator and the designer of the automated system may be responsible for the outcome of the triggering of a function. In addition to automation and autonomy, Bradshaw et al. [6] highlight another concept that is related to the self-sufficiency of the entity (defined as the French word for autonomous “autonome”) and that is the ability to take care of itself. An entity with a high level of self-sufficiency should be given the authority on the functions that are related to the acquisition of its required resources. Allocation of functions is a pillar of automation design. Parasuraman et al. have defined a classification of different Levels of Automation (LoA) [22]. These LoA have been extensively used for assessing automation levels of command and control systems such as Air Traffic Management applications, aircraft cockpits or satellite ground segments. As none of these systems reach level 10 (full automation), they are usually called human in-the-loop command and control systems of partly autonomous systems [18]. Beyond that, these LoA were also used as a design driver for research and industry projects having as a target higher automation levels[1]. However, we argue that authority and responsibility should also be taken into account at design time.

The three main aspects of automation at design time thus lay in describing what functions/tasks are allocated to the system and the human (allocation of functions), who is allowed to perform what functions/tasks (authority), and who is responsible for the outcome of the execution of the functions/tasks (responsibility). Because increasing/decreasing automation can have a huge impact on human performance, workload, team size and human error, there is a need for methods and tools to support the assessment of the impact of automation design (including the positioning with respect to the LoAs) in early stages of the development process. In this article, we highlight the benefits of having a notation making it possible to describe (with models), in a complete and unambiguous way, allocation of functions, authority and responsibility. We argue that a dedicated notation provides support during various stages of the design and development of an autonomous or partly autonomous interactive system. The proposed notation makes these abstract concepts concrete enough to provide means for the independent analysis of each of them.

Next section (Sect. 2) identifies what information is needed, at design time, to take into account the allocation of functions, authority and responsibility and defines those concepts. Section 3 presents a qualitative analysis of the classification of the Levels of Automation according to the concepts of allocation of functions, authority and

[1] See page 5: 17 projects and 13 PhD funded by SESAR Joint Undertaking towards higher automation levels in aviation http://www.sesarju.eu/sites/default/files/documents/events/sesar2020-20150504/3_SESAR2020_ER_Info_Day_FV_David_Bowen.pdf.

responsibility. These two sections highlight the fact that there is no available technique for describing the allocation of functions, authority and responsibility during the design of partly-autonomous systems. Section 4 presents the elements of notation for task models to provide support for the identification and representation of allocation of functions, authority and responsibility. Section 5 illustrates these elements of notation with the example of the Game of 15.

## 2 The Concepts of Allocation of Functions, Authority and Responsibility and How to Use Them for Automation Design

This section presents the results of a literature review on approaches for designing automation that take into account allocation of functions, authority and responsibility (referred to as AFAR in the remainder of the paper). It first highlights the information that needs to be taken into account when dealing with the allocation of functions, authority and responsibility. It also highlights that most of the related work focusses on the techniques for dealing with the allocation of functions, and does not provide precise guidance for taking into account authority and responsibility.

### 2.1 Allocation of Functions

The concept of *Allocation of Functions* refers to "determining the distribution of work between humans and machines early in the design process" [26]. Human work is the set of perceptive, cognitive, motor and input interactive tasks that the user should perform to reach her/his goal. System work is the set of algorithmic, input and output functions that the system should perform to support user goal. The analysis of the allocation of functions is necessary to identify the optimal distribution of both functions and tasks between a partly-autonomous system and a user. The allocation of functions is also central to the design of automation because it provides support to migrate user activities to be performed by the system or to migrate system functions to be performed by the user. Indeed, according to [27], not enough functions allocated to user will lead to underload and boredom and thus decreased performance while too many functions will lead to cognitive, perceptive or motoric overload and increase stress and likelihood of user errors. The output of the allocation of functions is the description of the sets of tasks that the user should perform to reach her/his goal and the description of the sets of functions that the system should perform to support user goal. This implies that the system designer has to identify all the functions that have to be performed by the system together with all the tasks that have to be performed by the user.

### 2.2 Authority

The concept of *Authority* refers to "the power or right to give orders, make decisions, and enforce obedience" according to [21]. When dealing with the design of automation, Flemisch et al. [11] propose to refine this definition to "what the actor is *allowed* to do or not to do". Taking into account authority at design time requires analyzing how the authority can be shared between the user and the system, which possibly involves alternating between the user and the system over time. For instance, allowing the system to trigger a function on its own increases the overall system authority and decreases the one of the user. Thus, the goal of the design and analysis of the allocation of authority is to identify the optimal distribution of authority between an autonomous or partly-autonomous system and a user which heavily depends on the type of system considered (e.g. safety critical systems). The output of the authority distribution is the description of what the system and the user are allowed to do (and in particular, what functions the system will be authorized to perform and to trigger – also called *initiative*). This implies that the system designer has to identify and describe both the tasks that the user is allowed to perform and the functions that the system is allowed to perform.

Going back to the definition of authority presented above, “the right to give orders” and “the right to enforce obedience” are already taken into account (for instance in a task description) when describing the functions and tasks that the system and the human are allowed to perform. However, in conformance with [14], we consider that the identification of “the right to make decisions” has to be done explicitly because there are complex relationships between decision-making authority and the allocation of functions and tasks between the system and the human.

## 2.3 Responsibility

The concept of *Responsibility* refers to the fact that an actor should be accountable for the result of an action [11]. The allocation of responsibilities (between the user and the system) must make explicit the outcomes that are relevant and who (the user or the system) influences these outcomes. The purpose of making responsibilities explicit is to be able to support the identification of the actor who has been at the root cause of an unwanted or unexpected outcome. The output of the allocation of responsibilities consists in a list of both all expected and all actual outcomes when an activity is performed. The comparison between actual outcomes and expected outcomes makes it possible to identify deviations (that could be errors on the user side or failures on the system side).

## 2.4 Related Work Addressing Allocation of Functions, Authority and Responsibility

Existing approaches dealing with automation design usually focus on identifying functions that should be allocated to either the operator or the system. These approaches provide support for the identification of which tasks are good candidate for automation and which ones should remain performed by the operator [4, 8, 10, 23, 26]. All of these approaches use task description techniques and provide support for describing the possible workflows between user tasks and system functions. In addition to the description of the possible workflows between user tasks and system functions, the concept of orchestration, as defined in the software engineering and business process modeling (BPM) literature [20], provides supports to describe the control over the possible workflows between user tasks and system functions. Orchestration models, usually represented with UML diagrams or BPM models, thus provide support to describe the initialization, the changes and the finalization between the workflows of the user and the system. Rovatsos et al. [24] highlighted the benefits of having an "orchestration workflow layer" in addition to descriptions of user tasks and system functions when developing systems that are able to adapt to different user behaviors.

Beyond analyzing allocation of functions and possible workflows at design time, Loer et al. [16] propose a model-checking technique to verify the relevance of all possible temporal scheduling (workflows) of adaptive automation.

We have found limited related work dealing with design approaches that provide support for taking into account authority sharing and responsibility issues. Gombolay et al. [14] discusses the observations they have made about decision-making authority and responsibility sharing between human and robots from the point of view of reaching a global human-robot optimized performance. Flemisch et al. [11] as well as Miller and Parasuraman [19] proposed a conceptual framework that highlight the importance of taking into account authority and responsibility at design time. Boy [5] proposed a conceptual model to support the analysis of authority sharing amongst several humans and systems. Cummings and Bruni [7] proposed to extend Parasuraman information processing model by adding a decision making component. However, none of this work provide precise techniques or even guidance to apply to describe or design the allocation of authority and/or responsibility between a system and its user.

## 3 Levels of Automation and Allocation of Functions, Authority and Responsibility (AFAR)

This section aims at discussing how the classification of the Levels of Automation, as defined by Parasuraman et al. [22], provides support for the design of the allocation of functions, authority when developing partly-autonomous systems. Table 1 presents the qualitative analysis of the Levels of Automation according to the allocation of functions, authority and responsibility (AFAR). The first column presents the Levels of Automation (LoA) as defined in [22], ranging from the highest automation (Level 10), where the "computer decides everything, acts autonomously, ignoring the human", to the lowest automation (Level 1), where "computer offers no assistance: human must take all decisions and actions". The second column presents the allocation of functions, authority and responsibility according to the description of the LoA.

Table 1 shows that going higher in automation levels affects Authority, sometimes Responsibility, or Allocation of Functions but that it is not done in a consistent way. Indeed, one could have expected that going from bottom to top AF, A and R would move progressively from user to computer. However, at LoA 7, the authority is already all to the computer, while it is shared with the human at LoA 8, and is again all to the computer at LoA 9. The same holds for the allocation of functions as, for instance, the user has to perform more actions at LoA 8 (perceive and ask for information) than at LoA 7 (where the user can only perceive information). In addition, even though most of the levels of automation concern partly-autonomous systems where both user and

**Table 1.** Levels of Automation (LoA) from [22] and its interpretation using AFAR

| Description of LoA as in [22] | Interpretation in terms of allocation of functions, authority and responsibility |
|---|---|
| 10. The computer decides everything, acts autonomously, ignoring the human | AF: All to computer<br>A: All to computer<br>R: All to computer |
| 9. Informs the human only if it, the computer, decides to | AF: All to computer but human might perceive the information presented<br>A: All to computer<br>R: All to computer |
| 8. Informs the human only if asked, or | AF: All to computer but human might ask and perceive the information presented<br>A: All to computer but human can ask to be informed<br>R: All to computer |
| 7. Executes automatically, then necessarily informs the human, and | AF: All to computer but human can perceive the information presented<br>A: All to computer<br>R: All to computer |
| 6. Allows the human a restricted time to veto before automatic execution, or | AF: All to computer but human can perceive how to veto and trigger veto<br>A: Mostly to computer but human can take authority over computer using veto<br>R: To computer if no veto and to human if veto |
| 5. Executes that suggestion if the human approves, or | AF: All to computer but human can perceive suggestion as well as how to approve and to trigger approval/denial<br>A: Mostly to computer but human can take authority by rejecting suggestion<br>R: Shared if approval and to human if not approved |
| 4. Suggests one alternative | AF: All to human but computer must compute and present one alternative<br>A: Mostly to human, computer can only provide suggestion<br>R: Shared if human selects one element of the options presented |
| 3. Narrows the selection down to a few, or | AF: All to human but computer must filter out and present options<br>A: Mostly to human, computer can only filter out options<br>R: Shared if human selects one element of the options presented |
| 2. The computer offers a complete set of decision/action alternatives, or | AF: All to human but computer must present the complete set of options<br>A: All to human<br>R: All to human |
| 1. The computer offers no assistance: human must take all decisions and actions | AF: All to human but computer might allow human to provide input<br>A: All to human<br>R: All to human |

system are involved, they do not provide any information about the user interface and its associated interaction techniques. This is an important limitation of that LoA framework as partly-autonomous systems may embed complex UIs and that the tasks of interacting with this type of systems should be part of the description of the allocation of functions. Finally, this classification represents only abstract information about the allocation of functions and tasks. It is not sufficient when designing a partly-autonomous system, because quantitative data about user tasks and system functions, meaning precise descriptions of tasks and functions, is required to specify the behavior of the system, its UI and the operations that are allowed with it.

We choose the Parasuraman LoA [22] as an example because these LoA are widely used in the industry. Nonetheless, the same qualitative analysis can be done on other existing classifications of LoA. The interested reader can find more information in the Vagia et al. [25] literature review of proposed LoA. The Table 2 presents the results of the qualitative analysis of the levels of driving automation according to the allocation of functions, authority and responsibility (AFAR). This example focuses on the level 2 of the 6 levels of driving automation proposed by the standard SAE J3016 [15] for the design of automated cars.

**Table 2.** Levels of driving automation from [15] and its interpretation using AFAR

| Description of level of driving automation as in [15] | Interpretation in terms of allocation of functions, authority and responsibility |
|---|---|
| 2. Partial Driving Automation<br>Human Driver (at all times):<br>• Performs the remainder of the Dynamic Driving Task not performed by the driving automation system<br>• Supervises the driving automation system and intervenes as necessary to maintain safe operation of the vehicle<br>• Determines whether/when engagement and disengagement of the driving automation system is appropriate<br>• Immediately performs the entire Dynamic Driving Task whenever required or desired<br>Driving Automation System (while engaged):<br>• Performs part of the Dynamic Driving Task by executing both the lateral and the longitudinal vehicle motion control subtasks<br>• Disengages immediately upon driver request | AF: Mostly to the human driver, the human driver can delegate the dynamic driving function to the driving automation system<br>A: Mostly to the human driver, the driving automation system can trigger both the lateral and the longitudinal vehicle motion control subtasks if the human driver engaged the driving automation system<br>R: All to the human driver, except for the lateral and longitudinal movements if the human driver engaged the driving automation system |

At this level of driving automation, all the functions are allocated to the human driver. However, the human driver can decide to delegate lateral and longitudinal motion control subtasks to the driving automation system. Then, the human driver has all the authority and responsibility except if the human driver decides to engage the driving automation system. In the end, this classification represents only abstract information about the responsibility and authority distribution between both entities.

Next section presents the elements of notation that aim at fulfilling the need of precise description of tasks and functions for the specification of the system.

## 4 Representing Authority, Responsibility and Allocation of Functions in Task Models

This section presents the task modeling elements of notation that aim at providing support for the explicit representation of allocation of functions, authority and responsibility. These elements of notations are demonstrated on the HAMSTERS notation but they could be added to other procedural descriptions of user tasks.

### 4.1 The Tool Supported Notation HAMSTERS

HAMSTERS (Human – centered Assessment and Modeling to Support Task Engineering for Resilient Systems) is a tool-supported task modeling notation for representing human activities in a hierarchical and structured way. The HAMSTERS notation and its eponym tool have initially been developed to provide support for ensuring consistency, coherence and conformity between user tasks and interactive systems at model level [1]. HAMSTERS embeds the common ground of task modelling such as hierarchical description of tasks, temporal ordering, refinement of tasks per types, manipulated data and structuring mechanisms [17]. This common ground is used to describe user tasks and system functions, and is thus used to describe the task and function allocation between user and system.

### 4.2 Allocation of Functions

The analysis of allocation of functions, authority, and responsibility requires at least one task model per role. The concept of role in HAMSTERS refers to a set of goals and tasks (described in one or several task models) that can be attributed to one actor. An actor is defined by an entity that is capable of performing a set of tasks and that has several characteristics such as physical properties, level of knowledge, experience…. Examples of actor can be a player (user playing a game) or a software application on a computer (as shown in Fig. 1(b)). In order to show the allocation of functions between the user and the autonomous part of the system, we propose to describe autonomous system functions in HAMSTERS as well. To distinguish user tasks from autonomous system functions in tasks models, we propose to describe autonomous system functions associated to corresponding dedicated roles in HAMSTERS. Figure 1(a) shows a typical example of project for studying automation in a mono-user computer game. The role called "Player as challenger" and the role called "Player as leader" contain respectively all the task models describing how the user can play as a challenger and as a leader. The role called "Player as game configuration manager" contain the task models describing how the user may configure the game if s/he is in charge of it. The role called "Software application as challenger" and the role called "Software application as leader" contain respectively all the task models describing how the software application can play as a challenger and as a leader. The role called "Software application as game configuration manager" contains the task models describing how the software application can perform tasks to configure the game (such as choosing the leader). The role "Software application as configuration maker" contains tasks models

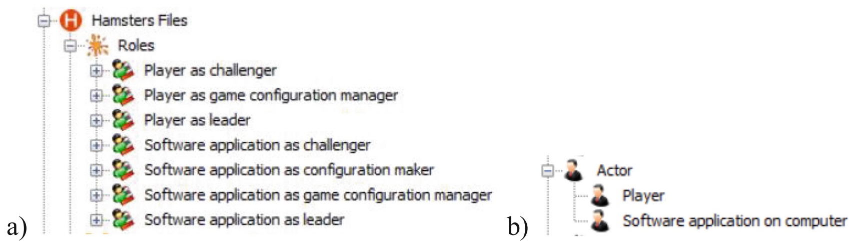

a)
Hamsters Files
Roles
Player as challenger
Player as game configuration manager
Player as leader
Software application as challenger
Software application as configuration maker
Software application as game configuration manager
Software application as leader
b)
Actor
Player
Software application on computer

**Fig. 1.** (a) The six mandatory roles for describing automation and (b) example of two actors

describing how the software application can provide support to the user to configure the game (as for example its tasks to record the winner for each game).

It is important to note the difference with standard task modeling practices where a user task model integrates in a single model an interleaving of user behavior and system's response to user behavior. In our case, in order to describe explicitly user tasks and autonomous system functions, we require the creation of several roles and at least one associated task models for each role. This means that an autonomous system model (belongs to the one of the roles that can be attributed to the software application) is only made of tasks belonging to the system task category, while the user model (here belongs to one the roles that can be attributed to the player) can embed any type of task type but system tasks. However, due to this separation of concerns, it is impossible to describe interleaving of actions between user and system inside those models (as it is usually done in task modelling notations). For this reason, we have added a new event-based mechanism dedicated to the explicit description of interleaving of actions between the user and the system. Figure 2 (resp. Fig. 3) presents an example of software application task model (resp. player task model). These task models contain description of the events (grey boxes) that are produced (an outgoing arrow from a task to an event) in one task model and that trigger tasks (an incoming arrow from an event to a task) in the other task model.

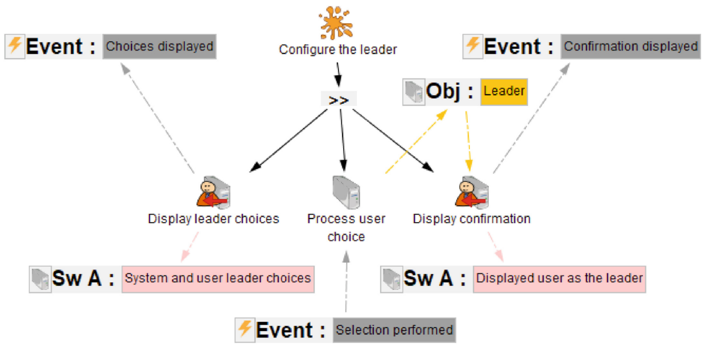

Event : Choices displayed
Configure the leader
Event : Confirmation displayed
Obj : Leader
>>
Display leader choices
Process user choice
Display confirmation
Sw A : System and user leader choices
Sw A : Displayed user as the leader
Event : Selection performed

**Fig. 2.** Task model describing the software application behavior related to the task model of the player (see Fig. 3) for choosing the leader

For example, in the software application task model depicted in Fig. 2, the output interactive system task “Display leader choices” produces the event “Event: Choices displayed”. This event triggers the execution of the visual perception task “See possible choices” in the player task model depicted in Fig. 3. Still in the player task model in Fig. 3, once the player has chosen who will be the leader (cognitive tasks under the temporal ordering operator choice “[]”), the player performs the selection (interactive input task “Select the leader”). This interactive input task produces an event “Event: Selection performed”), which is described in the software application task model (depicted in Fig. 2) as triggering the system task “Process selection”.

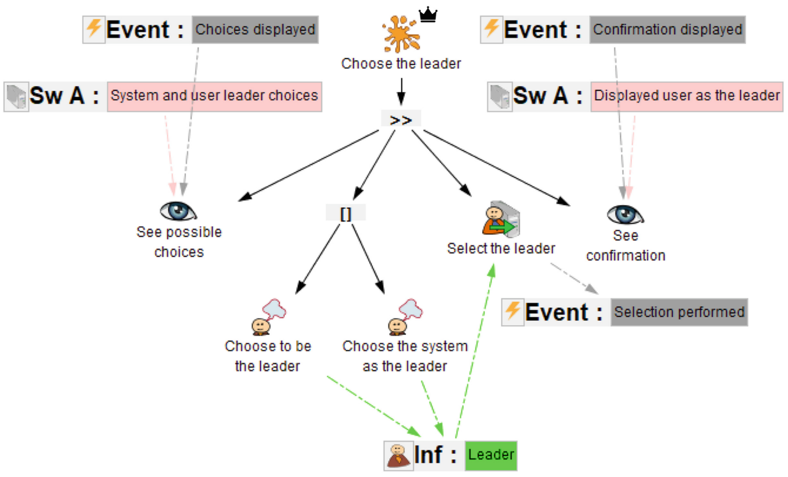

Event : Choices displayed
Choose the leader
Event : Confirmation displayed
Sw A : System and user leader choices
Sw A : Displayed user as the leader
>>
See possible choices
[]
Select the leader
See confirmation
Event : Selection performed
Choose to be the leader
Choose the system as the leader
Inf : Leader

**Fig. 3.** Task model describing the behavior of the player to choose who (between software application and player) is the leader – this model is triggered by the model in Fig. 2

The description of the orchestration of the workflows of the user tasks and system functions is described in an orchestration model. The elements of notation used to describe the orchestration are the same than the HAMSTERS elements of notation with adding two icons. The first one is the icon “conductor” (depicted in Fig. 4(a)).

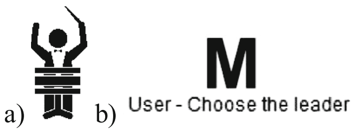


**Fig. 4.** (a) Symbol of the root node of a model describing how a set of task models is orchestrated, and (b) Representation of an entire task model that is used inside an orchestration model

The second one, is the icon “Model” (depicted in Fig. 4(b)), that is to be used in an orchestration model to represent a task model that can be started. In the orchestration model the temporal ordering of task models is represented using the standard operators in HAMSTERS. Comparing to swim lines with one possible sequence in BPM notations, the temporal ordering operators provide support for describing several distinct possible orchestrations. The orchestration model is used to describe the initialization of the workflows (which tasks and functions are started on the user side and on the system side), the possible dynamic changes (for example to represent that a function that is delegated to the system but that it could be re-assigned to the user) and the final completion of the workflows of the user and the system (what are the last user tasks and system functions).

## 4.3 Authority

The description of Authority can be either procedural or declarative. The procedural description of tasks aims at making what the system is allowed to do and what the user is allowed to do. It provides support to describe how authority goes from one role to the other one. In HAMSTERS this is represented using the event-based description of the triggering of tasks in another model. This view describes the switching of authority between the user and the system while the tasks are executing.

The declarative description aims at explicitly highlighting which tasks represent the right to make decisions, in order to facilitate the sharing of the decision-making. The icon “crown” (depicted in Fig. 5(a)) provides support to describe a task for which the user or the system has the decision-making authority. Only the tasks of type “abstract” may be represented with the symbol “authority” (depicted in Fig. 5(b)). This is because the user or the system can have the authority on a decision-making task. The refinement of the task and its associated set of actions is independent from the fact of having decision-making authority on it.

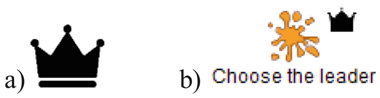


**Fig. 5.** (a) Symbol representing the authority (b) authority symbol associated to a task. The symbol is displayed when the property “authority” of a task is set to true.

Within this context of sharing authority, it is important to note that operators can perform actions by mistake or intentionally by violation. In the context of task descriptions, only nominal actions are represented as errors and deviations should be made explicit in other description [13]. Each task described in a task model means that the operator in charge of it is allowed to perform it.

### 4.4 Responsibility

The description of Responsibility can be either procedural or declarative. The procedural view is the description of the flow between the task and the outcome of the task. This description consists in a flow (arrow) between a task and information or objects describing the expected and actual results. The declarative view is the explicit representation of the symbol of "responsibility" in the relevant tasks, information and objects. The expected and actual results produced by a task can be represented with the data types "Information" (depicted in Fig. 6(a)) when the result is produced by the user, or with the data type "Object (depicted in Fig. 6(b)) when the result is produced by the system.

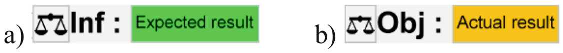


**Fig. 6.** Elements of the notation for representing responsibilities.

The tasks that set the expected results and/or have an impact on the actual results are tagged with the icon "scale". The icon "scale" (depicted in Fig. 7(a)) is displayed next to the user or system task (depicted in Fig. 7(b)). This icon represents that the user or the system is responsible for setting the expected results and/or for having an impact on the actual result.

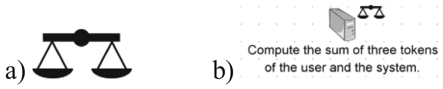


**Fig. 7.** (a) Symbol representing the responsibility (b) responsibility symbol associated to a task.

During the task modeling process, the task that produces the expected result has to be tagged with the symbol «responsibility». This symbol is displayed when the property "responsibility" of a task is set to true. These tasks have also to be connected to the information or object that describes the expected result, thanks to an arrow going from the task to the information or object describing the expected result. In addition, all the tasks that have an impact on the actual result have to be tagged with the symbol «responsibility». These tasks have also to be connected to the information or object that describes the actual result, thanks to an arrow going from the task to the information or object describing the actual result.

## 5 Illustrative Example: The Game of Fifteen

This section presents how the HAMSTERS notation supports the description of the allocation of functions, authority and responsibility through a simple case study.

## 5.1 Game of Fifteen: Main Principles and Rules

The Game of Fifteen is a two players game in which each player chooses and selects, in turn, a number (graphically represented as a token) ranging from 1 to 9. The first player who gets a combination of three numbers (amongst the set of tokens that s/he has selected) for which the sum is exactly 15 wins the game. No explicit rule defines who plays first, thus requiring players to reach an agreement. In the computerized version of the game, the software application on computer may act as a referee. In that case, it could declare the winner as soon as one of the players' set of tokens matches the winning condition. An example of a user interface for a computerized version of this game is shown in Fig. 8.

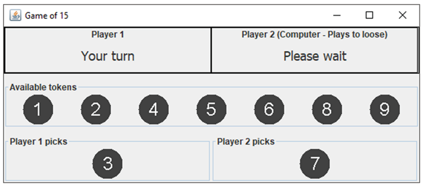
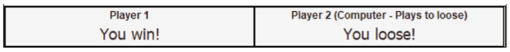
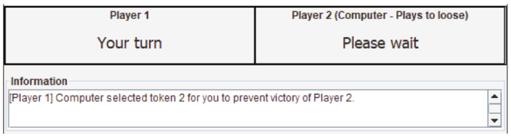
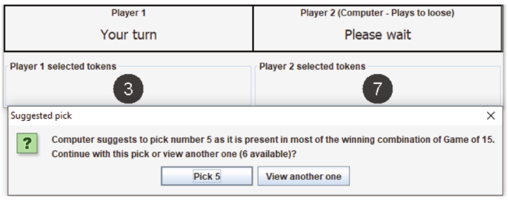
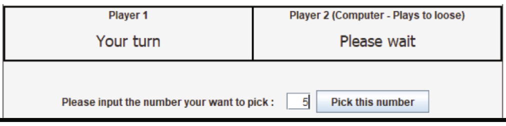


**Fig. 8.** An example of User Interface for the Game of Fifteen.

## 5.2 User Interfaces for the Game of 15 and Their Associated Levels of Automation

Table 3 presents several examples of user interface matching the definitions of levels of automation for the player task of selecting a token (number). We observe that at level 1 (LoA 1, Table 3), there is no information displayed regarding available or selected tokens. The player must memorize the already selected tokens before picking up an available one. Thus, at LoA 1, AFAR are all to human but the computer allows the human to provide inputs. At the highest level of automation (LoA 10, Table 3), there is no information displayed regarding available tokens either since the computer ignores the human for the fully automated task. Thus, AFAR are all to computer. From LoA 2 to LoA 6 (UI for LoA 2 and 5 are presented in Table 3) the player remains active in the decision process and is explicitly informed of what is going on via the User Interface. However, this UI evolves from "offering complete freedom" (at LoA 1) to "choose one token" (LoA 2). In those cases AFAR are all to human and the computer must present the entire set of options. At LoA 5, AFAR moves towards computer as the user must consider a suggestion made by the system. At this level (LoA 5), we observe the Allocation of Functions is all to computer but the human can perceive the computer's suggestion using the dialog window containing trigger for approval ("Pick 5") and denial ("View another one"). Authority is mostly to computer since it makes the suggestion, even though the human can take it back by denying the suggestion. Thus, responsibility is shared between human (if case of denial) and computer (if approval is granted).

**Table 3.** User interfaces matching the definition of Level of Automation for the game of 15.

| LoA | Definition [19] | Example of GUI supporting the LoA |
|---|---|---|
| 10 | The computer decides everything, acts autonomously, ignoring the human. |  Player 1 You win! Player 2 (Computer - Plays to loose) You loose! |
| 9 | [The computer] informs the human only if it, the computer, decides to. |  Player 1 Your turn Player 2 (Computer - Plays to loose) Please wait Information [Player 1] Computer selected token 2 for you to prevent victory of Player 2. |
| 6,7, 8 | Not presented due to space constraints | |
| 5 | [The computer] executes that suggestion [from LoA 4] if the human approves. |  Player 1 Your turn Player 2 (Computer - Plays to loose) Please wait Player 1 selected tokens 3 Player 2 selected tokens 7 Suggested pick Computer suggests to pick number 5 as it is present in most of the winning combination of Game of 15. Continue with this pick or view another one (6 available)? Pick 5 View another one |
| 3, 4 | Not presented due to space constraints | |
| 2 | The computer offers a complete set of decision/action alternatives. | See Fig. 8 |
| 1 | The computer offers no assistance: human must take all decisions and actions. |  Player 1 Your turn Player 2 (Computer - Plays to loose) Please wait Please input the number your want to pick : 5 Pick this number |

## 5.3 Modeling of Allocation of Functions, Authority, and Responsibility and with HAMSTERS

This section presents the description of the player tasks and of the software application tasks corresponding to different versions of the Game of 15. It aims at illustrating how the proposed notation provides support for the description and the analysis of the aspects related to AFAR during the design phases of a partly autonomous system. Due to space constraint, we have selected a set of representative models to illustrate all the proposed elements of notation.

The **allocation of functions** between the player and the software application is described in player task models (one of them is depicted in Fig. 10) and software application task models (one of them is depicted in Fig. 11). The **orchestration model** (depicted in Fig. 9) describes the possible orderings between the workflows of the software application tasks models and player tasks models. In this orchestration model, the player is in charge to choose the leader (represented by the model "Player as game configuration manager – Choose the leader" task model under the concurrent "|||" operator). Figure 3 depicts the user tasks of this model. Concurrently, the system provides a mean to configure the player choice of the leader (represented by the model “Software application as configuration maker – Configure the leader” task model with concurrent operator). Figure 2 depicts the player tasks of this model. The model “Software application as configuration maker – Configure the leader” produces the object “Leader” (system side) and the model “Player as game configuration manager - Choose the leader” produces the information “Leader” (user side). Both elements of data contain the reference to the name of leader of the game. Then, if the user is the leader (left branch under the choice “[]” operator in Fig. 9), s/he starts to play as the leader (condition on the information “leader”) and the software application on computer starts to play as the challenger (condition on the object “leader”). Alternatively, if the player is the challenger (right branch under the choice “[]” operator in Fig. 9), s/he starts to play as the challenger (condition on the information “leader”) and the software application on computer starts to play as the leader (condition on the object “leader”). Finally, the software application is in charge to store the winner at the end of the game (last model “Software application as configuration maker – Store the winner” on the right under the sequence “≫” operator).

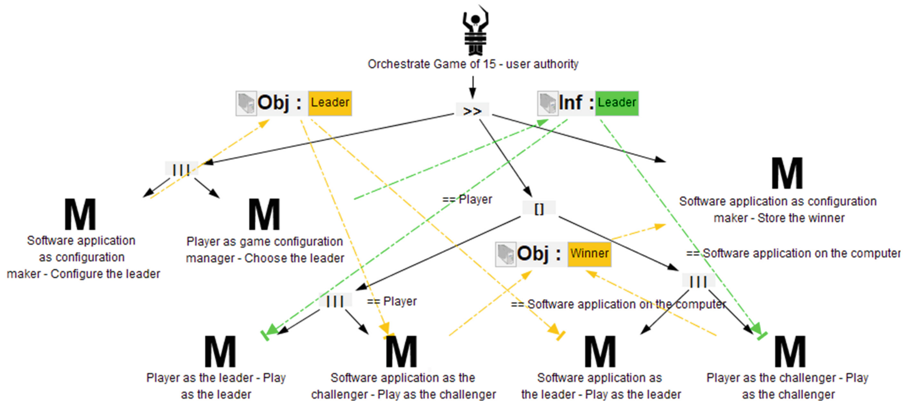

Orchestrate Game of 15 - user authority
Obj : Leader
>>
Inf : Leader
III
Software application as configuration maker - Configure the leader
Player as game configuration manager - Choose the leader
== Player
[]
Software application as configuration maker - Store the winner
Obj : Winner
== Software application on the computer
III
== Player
III
== Software application on the computer
Player as the leader - Play as the leader
Software application as the challenger - Play as the challenger
Software application as the leader - Play as the leader
Player as the challenger - Play as the challenger

**Fig. 9.** Orchestration model of the computerized version of the Game of Fifteen.

Figure 10 depicts the **software application task model** of the software application tasks that have to be performed to play a turn of the version of the Game of 15 for the LoA 5. In order to process user turn (abstract task “Process user turn (level 5)” in Fig. 10), the system sequentially (sequence “≫” operator): displays tokens played by both the players and a suggested token for the user (interactive output tasks “Display tokens played by user” and “Display tokens played by system”). Then, the system suggests and displays a token iteratively (abstract iterative task “Suggest tokens”) on user demand until the user confirms one of the suggested token (system task “Process user confirmation” under disable “[>” operator). The abstract iterative task “Suggest tokens” consists in the following actions. The system has to suggest a token that have to help the user to win (system task “Suggest a token” that accesses to the declarative knowledge “The token suggested have to help the user to win”). Then, the system displays the suggested token through a pop-up window (software application information "Suggested pick pop-up") and triggers an event ("Suggested token is displayed" event). The system cannot execute its following tasks ("Suggest a token" and "Process user confirmation" system tasks have an input event) until one of the two user events are triggered: user clicks on "View another one" button (interactive input task "User clicks on "View another one" button" in Fig. 11) or user clicks on "Pick [number] button" (interactive input task "User clicks on "Pick [number]" button" in Fig. 11).

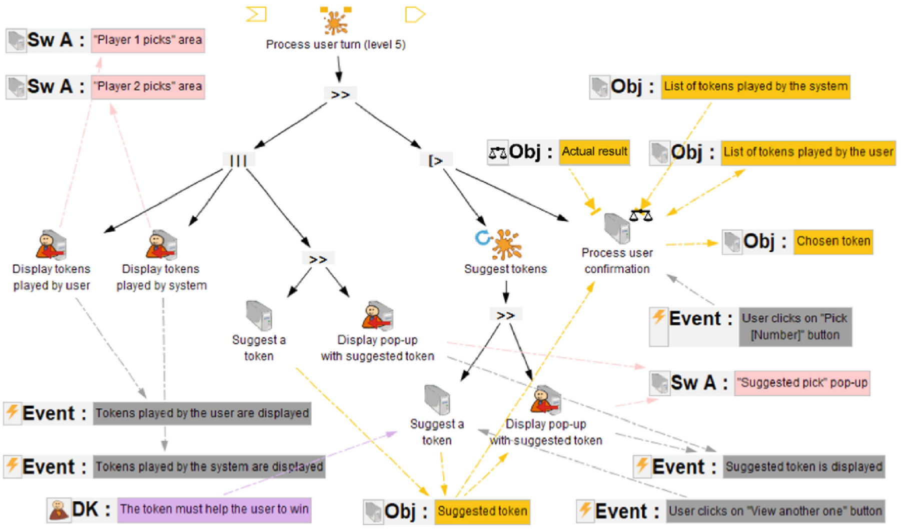

Sw A : "Player 1 picks" area
Sw A : "Player 2 picks" area
Process user turn (level 5)
>>
|||
[>
Obj : List of tokens played by the system
Obj : Actual result
Obj : List of tokens played by the user
Display tokens played by user
Display tokens played by system
>>
Suggest a token
Display pop-up with suggested token
Suggest tokens
Process user confirmation
Obj : Chosen token
>>
Event : User clicks on "Pick [Number]" button
Sw A : "Suggested pick" pop-up
Suggest a token
Display pop-up with suggested token
Event : Tokens played by the user are displayed
Event : Tokens played by the system are displayed
Event : Suggested token is displayed
DK : The token must help the user to win
Obj : Suggested token
Event : User clicks on "View another one" button

**Fig. 10.** Task model of the software application task "Process user turn" (for LoA 5).

Figure 11 depicts the **player task model** of the tasks that have to be performed by the player to play a turn of the version of the Game of 15 for the LoA 5. For this LoA, the player supervises the choice of token (abstract task "Supervise token choice" task in Fig. 11). In that figure, the player analyzes and reads the token suggested by the system until s/he confirm one (iterative abstract "Analyze and read suggestion" under a disable "[>" operator with abstract task "Confirm suggested token"). At any time, the player can think about the token to play during the supervision (iterative and optional cognitive task "Think about the token to play" under the concurrent "|||" operator). More precisely, the event "Suggested token is displayed" that is triggered by the system allows the user to see the suggested token (motoric sight task "See suggested token") and to memorize it (cognitive task "Memorize the suggested token"). In the same way, the player can read the tokens played by the system and by her or him in an independent order (order independent "|=|" operator between user tasks "Know suggested tokens" and "Know tokens played by the system"). Then, the player analyses the suggested token and she or he decides to select the suggested token or to ask for another one (sequence of cognitive decision task, motor task and user input task).

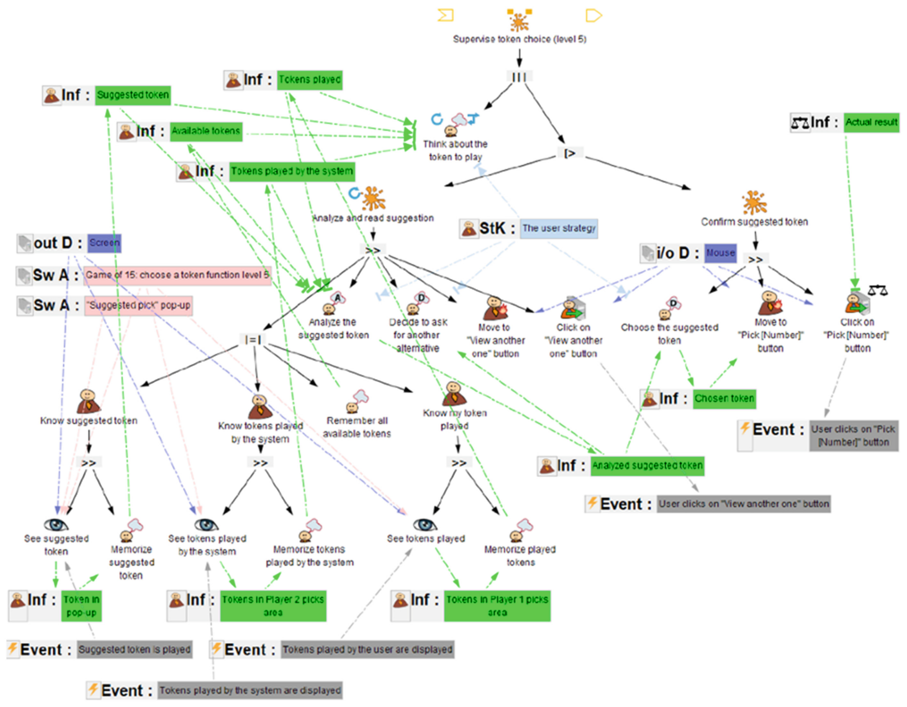

Supervise token choice (level 5)
Inf : Suggested token
Inf : Tokens played
Inf : Available tokens
Inf : Tokens played by the system
Think about the token to play
Inf : Actual result
Analyze and read suggestion
StK : The user strategy
Confirm suggested token
out D : Screen
Sw A : Game of 15: choose a token function level 5
Sw A : "Suggested pick" pop-up
i/o D : Mouse
Analyze the suggested token
Decide to ask for another alternative
Move to "View another one" button
Click on "View another one" button
Choose the suggested token
Move to "Pick [Number]" button
Click on "Pick [Number]" button
Know suggested token
Know tokens played by the system
Remember all available tokens
Know my token played
Inf : Chosen token
Event : User clicks on "Pick [Number]" button
Inf : Analyzed suggested token
Event : User clicks on "View another one" button
See suggested token
Memorize suggested token
See tokens played by the system
Memorize tokens played by the system
See tokens played
Memorize played tokens
Inf : Token in pop-up
Inf : Tokens in Player 2 picks area
Inf : Tokens in Player 1 picks area
Event : Suggested token is played
Event : Tokens played by the user are displayed
Event : Tokens played by the system are displayed

**Fig. 11.** Task model of the player task "Supervise the token choice" (LoA 5).

The input and output events between user tasks and system tasks describe the procedural change of **authority** between the player and the software application. An output event from a system task in conjunction with an input of this event to a user task describes a switch of authority from the system to the user. An output event from an interactive input task in conjunction with and input of this event to a system task describes a switch of authority from the player to the software application. However, even if one of the roles has the authority on the other for a task, the other can execute other concurrent task over which s/he has the authority like thinking about the token to play (cognitive task "Thinking about the token to play" in Fig. 11). In this version of the Game of 15, there is one task related to **decision-making authority**, it is the task "Choose a leader" and the player has the authority on it (as depicted in Fig. 2). The explicit representation of the authority on this decision task provides support for discussing about what would be the impact if assigned to the software application, or if transferred from the player to the software application at runtime. When the player confirms a suggested token, the software application has the **responsibility** to process correctly the suggested token confirmed by the player (system task "Process user confirmation" in Fig. 10) and the player has the responsibility to confirm the correct suggested token according to his or her expected result ("Click on "Pick [number] button" input task in Fig. 11). Both tasks have an impact on the outcome of the game (connection between these tasks and the corresponding objects and information in

Figs. 10 and 11). The explicit representation of responsibility by the description of the expected and actual outcome of these tasks provide support for arguing about the actions that should be taken if the user or the system tasks fail in reaching the expected outcome (e.g. modifying the display size of the tokens and or buttons if the user do not "Click on "Pick [number] button").

## 6 Future Work

We have presented extensions to a task-modelling notation and tool in order to provide support to the explicit description of allocation of functions, authority and responsibility. As a future work, we plan to propose an approach for the qualitative analysis of allocation of functions, authority and responsibility based on task models. For example, the systematic analysis of the required cognitive tasks described in the task models as well as the information manipulated by the human for different allocation of functions, authority and responsibility distributions will aim at providing insights on the impact of these choices of allocation on the cognitive workload. Another example of analysis that would be part of the approach is the analysis of motor tasks described in in the task models to determine the impact on the effort for example. Furthermore, the analysis of the number of tasks and the types of tasks allocated to the human can provide a model-based analysis of some user experience aspects. This type of analysis can help to prevent design solutions where the human can be bored or complacent in case of high-level of automation. This approach will aim at providing a comparison between different distributions of allocations of functions, authority and responsibility.

Another possible future work is to introduce the description of possible errors in the task models (this technique is already supported by HAMSTERS notation and tool [13]) to provide support to analyse how to give back the authority to the human in case of automation failure.

## 7 Conclusion

Automation has been studied for many years and even though metaphors [12] or frameworks [22] have been proposed, the description of the allocation of functions, authority and responsibility between the user and the system is not supported by notations and tools. However, when designing automation, a precise description of those elements are required in order to:

(1) identify and specify the partly-autonomous system functions and the user tasks,
(2) identify and reason about the actions the system is allowed to trigger and the decisions the system is allowed to take, (the similar holds on the user side),
(3) understand mutual responsibility (and liability) in case the cooperation between the user and the partly-autonomous system does not produce the expected outcomes.

Existing approaches for the design of automation mainly focus on the allocation of functions and deal with authority and responsibility only at a high abstraction level.

This does not provide support for reasoning about the quality of a given allocation of authority and responsibility and makes the task of engineering of partly-autonomous system cumbersome, leaving design decisions in the hands of the programmers. This article has argued that the analysis of the allocation of functions must go beyond the analysis of the sharing of the tasks of high-level types (decision, suggestions, commands as proposed in [22]) and that fine-grain descriptions of user and system actions are required. This article also argued that the allocation of authority and responsibility has to be taken into account at the same fine-grain level as the allocation of functions and tasks.

We have proposed several extensions to an existing notation for describing user tasks in order to make it possible to represent in an explicit manner these three elements. We have demonstrated on a case study that the extended notation makes it possible to describe these three elements on a concrete example and that these descriptions provide complementary information with respect to the Levels of Automation classical approach for automation design. Future work will be dedicated to the use of this notation at design time to design function allocation between the system and the user in order to avoid the pitfalls exhibited by [27] and build systems that support best operators in their tasks.

However, in a similar way as human can make errors, automation can fail and asking user to take over is not a viable option [1]. In order to ensure continuity of service, the automation should degrade in a graceful way, reconfiguring itself as this can be done with interactive or classical systems [3]. Such dynamic reconfigurations raise interesting and challenging issues that are not covered by the presented approach but will be addressed in future work. Finally, system behavior description might require more powerful notations (for instance making explicit large number of states) than the one of HAMSTERS. In order to address this, the use of complementary and compatible notations will be required as proposed in [2].

## References

1. Bainbridge, L.: Ironies of automation. Automatica **19**, 775–780 (1983)
2. Barboni, E., Ladry, J-F., Navarre, D., Palanque, P., Winckler, M.: Beyond modelling: an integrated environment supporting co-execution of tasks and systems models. In: Proceedings of EICS 2010, pp. 143–152. ACM
3. Basnyat, S., Navarre, D., Palanque, P.: Usability service continuation through reconfiguration of input and output devices in safety critical interactive systems. In: International Conference on Computer Safety, Reliability and Security (SAFECOMP 2008), Newscastle, UK (2008)
4. Boy, G.: Cognitive function analysis for human-centered automation of safety-critical systems. In: Proceedings of ACM CHI 1998, pp. 265–272 (1998)
5. Boy, G.: Orchestrating situation awareness and authority in complex socio-technical systems. In: Aiguier, M., Caseau, Y., Krob, D., Rauzy, A. (eds.) CSDM 2012, pp. 285–296. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-34404-6_19
6. Bradshaw, J.M., Hoffman, R.R., Woods, D.D., Johnson, M.: The seven deadly myths of “autonomous systems”. IEEE Intell. Syst. **28**(3), 54–61 (2013)

7. Cummings, M.L., Bruni, S.: Collaborative human–automation decision making. In: Nof, S. (ed.) Springer Handbook of Automation, pp. 437–447. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-78831-7_26
8. 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)
9. Dictionary. English dictionary. www.dictionary.com/browse/automation. Accessed Sept 2018
10. Dittmar, A., Forbrig, P.: Selective modeling to support task migratability of interactive artifacts. In: Campos, P., Graham, N., Jorge, J., Nunes, N., Palanque, P., Winckler, M. (eds.) INTERACT 2011. LNCS, vol. 6948, pp. 571–588. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23765-2_39
11. 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. Technol. Work **14**(1), 3–18 (2012)
12. Flemisch, F., Adams, C., Conway, S., Goodrich, K., et al.: The H metaphor as a guideline for vehicle automation and interaction, NASA TM, 2003-212672 (1975)
13. Fahssi, R., Martinie, C., Palanque, P.: Enhanced task modelling for systematic identification and explicit representation of human errors. In: Abascal, J., Barbosa, S., Fetter, M., Gross, T., Palanque, P., Winckler, M. (eds.) INTERACT 2015. LNCS, vol. 9299, pp. 192–212. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22723-8_16
14. Gombolay, M.C., Gutierrez, R.A., Clarke, S.G., Sturla, G.F., Shah, J.A.: Decision-making authority, team efficiency and human worker satisfaction in mixed human—robot teams. Auton. Robots **39**(3), 293–312 (2015)
15. J3016 Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems SAE International (2014)
16. Loer, K., Hildebrandt, M., Harrison, M.: Analysing dynamic function scheduling decisions. In: Johnson, C.W., Palanque, P. (eds.) Human Error, Safety and Systems Development. IIFIP, vol. 152, pp. 45–60. Springer, Boston, MA (2004). https://doi.org/10.1007/1-4020-8153-7_4
17. Martinie, C., Palanque, P., Winckler, M.: Structuring and composition mechanisms to address scalability issues in task models. In: Campos, P., Graham, N., Jorge, J., Nunes, N., Palanque, P., Winckler, M. (eds.) INTERACT 2011. LNCS, vol. 6948, pp. 589–609. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23765-2_40
18. Martinie, C., et al.: Formal tasks and systems models as a tool for specifying and assessing automation designs (regular paper). In: 1st International Conference on Application and Theory of Automation in Command and Control Systems (ATACCS 2011), Barcelona, Spain. ACM DL, May 2011
19. Miller, C.A., Parasuraman, R.: Designing for flexible interaction between humans and automation: delegation interfaces for supervisory control. Hum. Factors **49**, 57–75 (2007)
20. Misra, J., Cook, W.R.: Computation orchestration: a basis for wide-area computing. J. Softw. Syst. Model. **6**(1), 83–110 (2007). https://link.springer.com/article/10.1007/s10270-006-0012-1
21. Oxford. English Dictionnary. https://en.oxforddictionaries.com/definition. Accessed Apr 2018
22. Parasuraman, R., Sheridan, T.B., Wickens, C.D.: A model for types and levels of human interaction with automation. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **30**(3), 286–297 (2000)
23. Pocock, S., Harrison, M.D., Wright, P.C., Johnson, P.: THEA: a technique for human error assessment early in design. In: INTERACT 2001, pp. 247–254 (2001)

24. Rovatsos, M., Diochnos, D.I., Wen, Z., Ceppi, S., Andreadis, P.: SmartOrch: an adaptive orchestration system for human-machine collectives. In: Proceedings of the Symposium on Applied Computing (SAC 2017), pp. 37–44. ACM, New York (2017)
25. Vagia, M., Transeth, A.A., Fjerdingen, S.A.: A literature review on the levels of automation during the years. What are the different taxonomies that have been proposed? Appl. Ergon. **53**, 190–202 (2016)
26. Wright, P.C., Dearden, A., Fields, B.: Function allocation: a perspective from studies of work practice. Int. J. Hum.-Comput. Stud. **52**(2), 335–355 (2000)
27. Yerkes, R.M., Dodson, J.D.: The relation of strength of stimulus to rapidity of habit-formation. J. Comp. Neurol. Psychol. **18**, 459–482 (1908)