	HoliDes **Holi**stic Human Factors **Des**ign of Adaptive Cooperative Human-Machine Systems	

D 3.5a - Techniques and Tools for Adaptation Vs1.5 incl. Handbooks and Requirements Analysis Update

Project Number:	332933
Classification:	Public
Work Package(s):	WP 3, T3.3, T3.4, T3.5
Milestone:	M3
Document Version:	V 1.0
Issue Date:	31.12.2014
Document Timescale:	Project Start Date: October 1, 2013
Start of the Document:	Month 19
Final version due:	Month 21
Deliverable Overview:	**Main Public document:** D 3.5a - Techniques and Tools for Adaptation Vs1.5 incl. Handbooks and Requirements Analysis Update , Public part **Confidential Document:** D 3.5b - Techniques and Tools for Adaptation Vs1.5 incl. Handbooks and Requirements Analysis Update , Confidential part **Annex I:** Communication Guidelines (public)
Compiled by:	Sara Sillaurren – TEC
Authors:	Martina Becker – EADS-IW-DE, Denis Javaux – SYM, Sara Sillaurren – TEC, Simona Collina – SVN, Roberta Presta – SVN, Richard Leblond (EAD-FR), Martin Boecker (EAD-CAS-DE), Stefan Griesche (DLR), Peter Van Der Meulen (PHI)
Reviewers:	Miriam Quintero – ATO Holger Prothmann – TWT
Technical Approval:	Jens Gärtner, Airbus Group Innovations
Issue Authorisation:	Sebastian Feuerstack, OFF

DISTRIBUTION LIST		
Copy type[1]	Company and Location	Recipient
T	HoliDes Consortium	all HoliDes Partners

[1] Copy types: E=Email, C=Controlled copy (paper), D=electronic copy on Disk or other medium, T=Team site (AjaXplorer)

RECORD OF REVISION		
Date	Status Description	Author
16.04.2015	Initial Version and structure	Sara Sillaurren (TEC) , Nicolas Schneider (EADS-FR), Richard Leblond (EADS-FR)
27.05.2015	Annex I: Communication Guidelines	Sara Sillaurren (TEC)
28.05.2015	HF Guidelines section	Martina Becker (EADS-IW-DE)
01.06.2015	Adaptation framework contribution	Denis Javaux (SYM)
03.06.2015	First Contribution round integration	Sara Sillaurren (TEC)
11.06.2015	Update for section 2, except communication guidelines	Richard Leblond (EAD-FR)
12.06.2015	Update for section 3.3 (Control Room)	Martin Boecker (EAD-CAS-DE) , Richard Leblond (EAD-FR)
15.06.2015	Update for section 2 Adaptation Framework	Richard Leblond (EAD-FR)
15.06.2015	Update for UC4	Stefan Griesche (DLR)
17.06.2015	Update for UC1	Peter Van Der Meulen (PHI)
25.06.2015	Internal review	Miriam Quintero (ATO) Holger Prothmann (TWT)
02.07.2015	Final version of document	Sara Sillaurren (TEC)

Table of Contents

Table of Tables

Table of Figures

1 Introduction

The objective of this public document *"D 3.5a - Techniques and Tools for Adaptation Vs1.5 incl. Handbooks and Requirements Analysis Update"* is to present the techniques and tools for adaptation, including context assessment, adaptation computation and communication/feedback.

In this document, Section 2 presents the Adaptation Framework updated from previous deliverable versions. After a recall of the concept of automation, it presents some elements about cooperative systems and characterizes the concept of adaptation. Going more in detail in the different components of the architecture relying on the notion of executive and adaptive loops, it indicates the main elements to be modeled and propose a first step to integrate this Architecture Framework in the HF-RTP.

An additional section has been included concerning HF guidelines to take into account before introduction of adaptation. It provides a detailed orientation for the development process of the AdCoS in HoliDes, considering the implementation of adaptive systems and Adaptive Automation (AA) into a cooperative multi-agent-system, including both humans and machines.

In Section 3, we present elements of this Adaptation Framework for four selected use cases (UC):

- UC1: Guided Patient Positioning (Section 3.1)
- UC2: Diversion Airport (Section 3.2)
- UC3: Command and Control Room (Section 3.3)
- UC4: Overtaking including lane change assistant (Section 3.4)

In the confidential document *"D 3.5b - Techniques and Tools for Adaptation Vs1.5 incl. Handbooks and Requirements Analysis Update"*, Section 2 investigates how common solutions can be found for context assessment, adaptation computation and communication/feedback between the four use cases. In Section 3, the output of Task 3.6 - *Integration of Techniques & Tools into the HF-RTP and Transfer for Application in WP6-9*, we will show how the available modules developed in Tasks 3.3, 3.4 and 3.5 will be implemented within the different UCs, the characteristics of the integration, the constraints and limits.

2 Adaptation Framework

In this section, we provide a description of a "Framework for Adaptation", in particular on the aspects that will allow characterizing each use case, in terms of context assessment, adaptation and communication.

2.1 Adaptation Framework Definition

In this section, we introduce the Adaptation Framework. The Adaptation Framework has been developed and described in *D3.3 - Framework for Adaptation*.

2.1.1 Motivation

As will be seen in the sections below, the Adaptation Framework brings new concepts to understand, model and design adaptation, Adaptive Systems, and more generally Adaptive Cooperative Human-Machine Systems (AdCoS).

The Adaptation Framework is based on the notion of control loop. Control loops are amongst the central tenets of modern science and technology.

Control - or feedback - loops have been found useful to understand and model complex dynamic systems, in particular intentional ones (i.e., systems with goals). Control loops can for example be found in biological systems (e.g., gene regulatory networks, physiology, hormonal regulations, etc.), ecological systems (e.g., population regulation), climate science, economics (e.g., stock markets modeling & investment strategies; economic equilibrium model), etc.

In the design and engineering realm, control loops are notably found and used in mechanical engineering (e.g. Watt's regulator), in electronic engineering and in all domains where trajectories have to be controlled (aviation, maritime, automotive, space). Control theory is one of the foundations of modern engineering.

More generally, control loops can be observed in most stable dynamic systems, where they are precisely the main mechanism through which stability is achieved. They are also the main mechanism allowing the achievement of goals in intentional systems. These apply to both natural and artificial systems, as shown above.

Adaptive cooperative human-machine systems (AdCoS) being artificial, intentional, and normally stable dynamic systems, it is not surprising to find control loops within AdCoS, and even to resort to them as the main central construct for designing AdCoS. In an AdCoS, control loops allow to maintain the system's stability and integrity, achieve goals through the performance of tasks on various types of processes (see Figure 9), provide cooperative behavior through coordinated interactions between the system's agents and finally adapt the whole system to changing circumstances.

2.1.2 Key concepts

2.1.2.1 Control loops

The Adaptation Framework is based on the notion of a control loop.

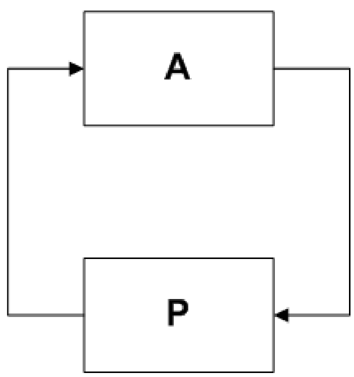

Figure 1: Control loop

An agent A closes a control loop on a process P to achieve specific goals. For example, a practitioner (A) positions a patient within a MRI device (P); a pilot (A) controls the trajectory of an aircraft (P).

Closing such a control loop involves several information processing steps, such as information gathering, decision-making and action (information-processing perspective); or perception, evaluation, decision-making, action planning and action implementation (cognitive perspective), see Figure 2.

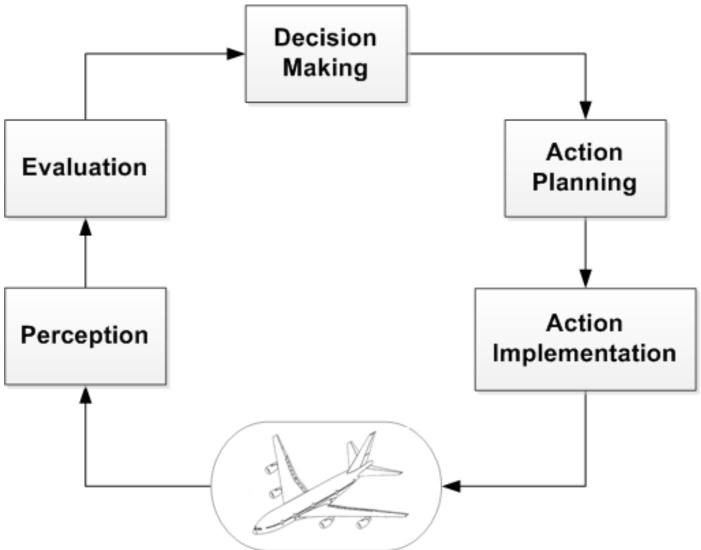

Figure 2: Control loop (cognitive perspective)

2.1.2.2 Executive and adaptive functions

A key idea of the Adaptation Framework is to dissociate between executive and adaptive functions, in a dual control loops architecture, see Figure 3.

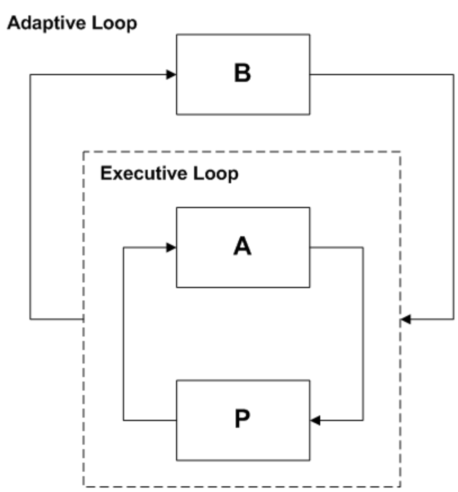

Figure 3: Adaptive loop on an Executive loop

An agent A closes an executive control loop on a process P and an agent B closes an adaptation control loop on the executive control loop achieved by A (including agent A).

This dissociation allows distinguishing between "what the agent does" (executive function) and the "adaptation of what the agent does" (adaptation function).

2.1.2.2.1 Characterization of the adaptation: what, when, how and why

When characterizing such an executive-adaptation control loop pair, the questions of what is adapted (what), in which circumstances (when), how (how), and why (why) also arise.

2.1.2.2.2 Context assessment, adaptation and communication

The notions of context assessment, (computation of) adaptation and communication are central to HoliDes. The Figure 4 explains what they relate to:

- context assessment occurs when the adaptive agent (B) takes information on the executive control loop
- adaptation - or computation of adaptation - occurs when B computes adaptations to perform on the executive control loop
- communication occurs when B alters how the executive control loop is performed.

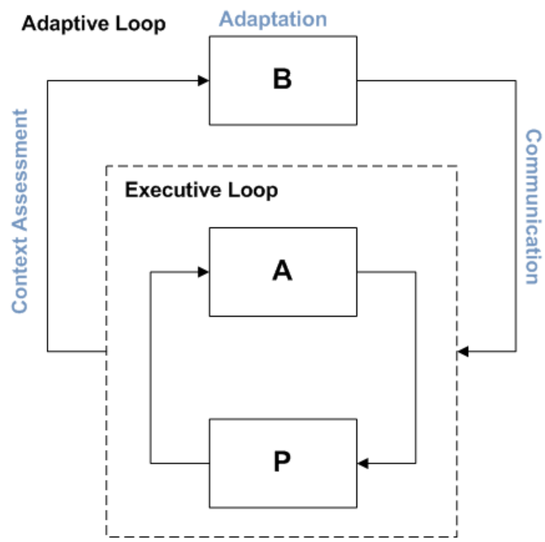

Figure 4: Context assessment, adaptation and communication

If the cognitive perspective on control loops (see Figure 2) is used, the overall picture for adaptive control loops is given in Figure 5. All steps, executive and adaptive, are seen as cognitive in nature.

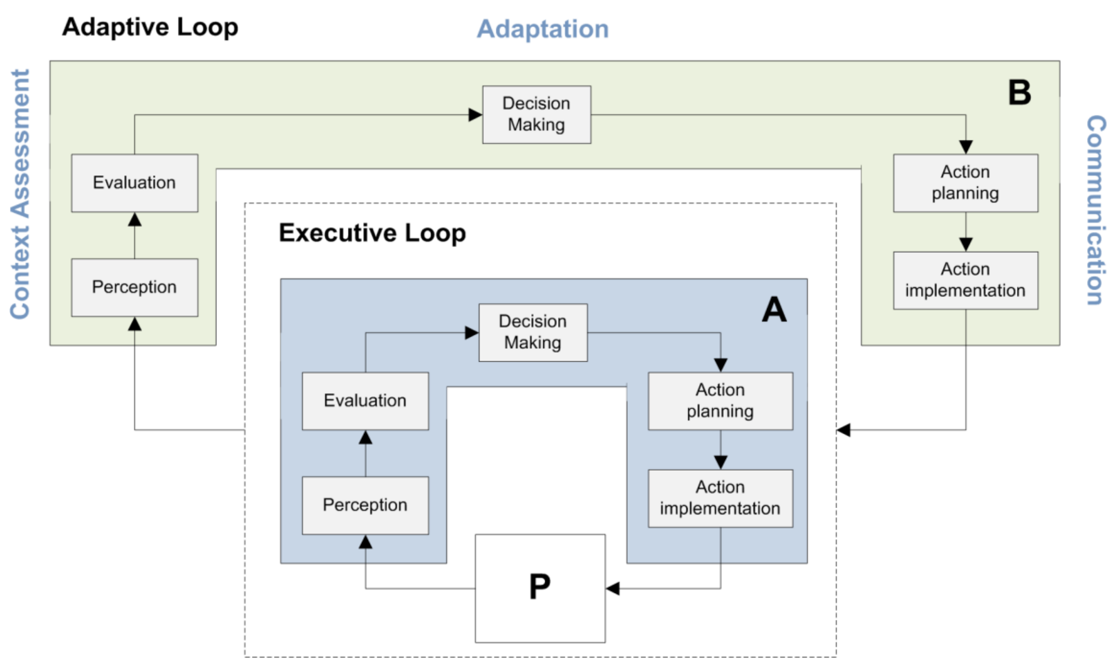

Figure 5: Adaptive control loop in the cognitive perspective

2.1.2.2.3 Relations between A and B

There are several possible schemes regarding how the executive and adaptive functions are allocated:

- **A=B**: In this case, the agent B is the agent A. This means that the executive and adaptation function are both performed by agent A. Agent A superposes the two functions. A is fully autonomous and self-adaptive.

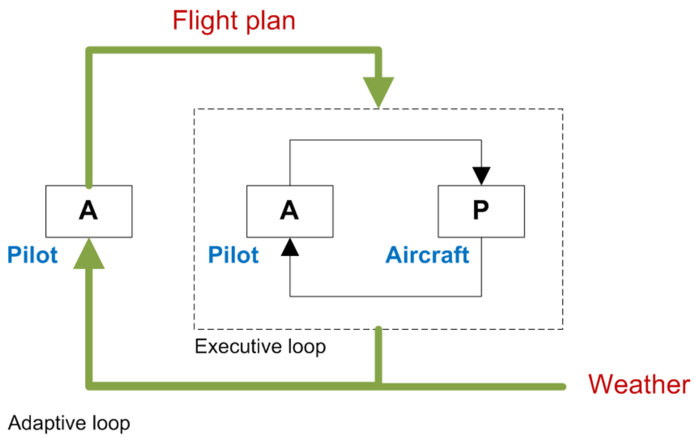

Figure 6: A=B

In Figure 6, the agent A is for example the pilot of a small aircraft. The pilot executes a pre-planned flight plan, with the intention to land at airport 1 (executive loop). Bad weather develops at airport1, the pilot acknowledges that state of affair, decides to land at airport 2 and adapt the flight plan accordingly (adaptive loop). Agent A executes both the executive and adaptive loops.

A here is a human agent. If A was a fully automated adaptive system (with for example weather recognition and decision-making capabilities), then agent A would resort from adaptive automation, combining executive and adaptive capabilities.

- **A<>B**: In this case, agent B and agent A differ. B and A are in a master-slave relationship (B master, A slave). B defines and adapts how A performs the executive functions.

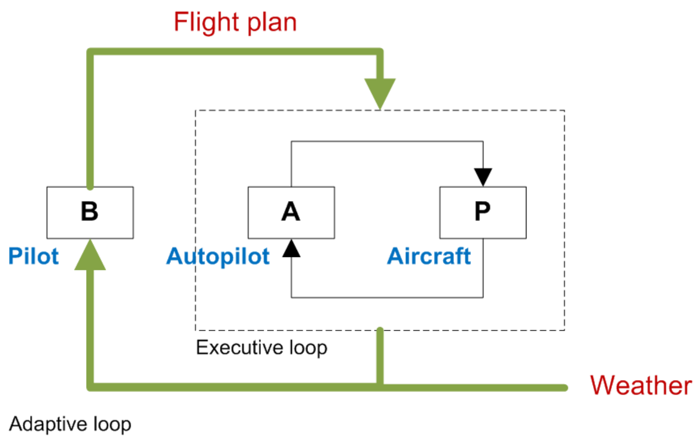

Figure 7: A<>B

In the figure above, the agent A is the autopilot of a small aircraft. The autopilot executes the pre-planned flight plan, defined by the pilot (agent B), with the intention to land at airport 1 (executive loop). When bad weather develops at airport 1, the pilot (agent B) alters the flight plan accordingly, to land at airport 2.

The relation between B and A is a typical master-slave relation, typical of non-adaptive automation. Adaptation (of automation) is achieved by the human agent.

- **A~B**: This covers all intermediary cases where A has some authority on how it performs its executive function.

  In Figure 8, the autopilot has some capability to influence, at least partially, the flight plan it is flying. For example, the autopilot - or better autoflight system here - through some flight management capabilities can process weather reports from destination airports and propose alternative flight plans to the pilot. It participates in the elaboration of an action plan it is itself executing.

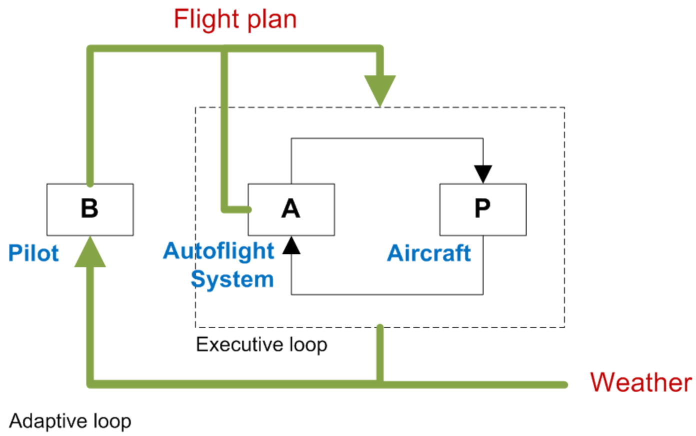

Figure 8: A~B

2.1.2.3 Agents and functions in cooperative human-machine systems

2.1.2.3.1 Cooperative human-machine systems

In HoliDes, a cooperative human-machine system is defined as a set of human and machine agents acting cooperatively on some controlled entity (process, plant, vehicle, etc), within a particular environment.

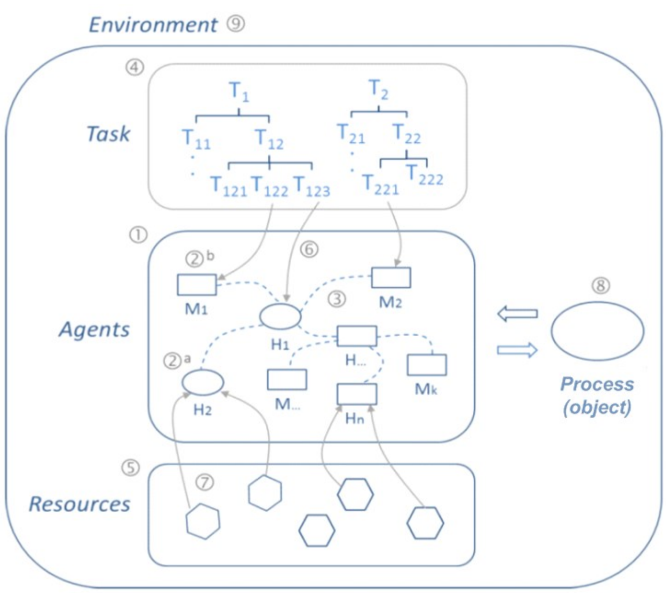

Figure 9: Cooperative System (CoS)

A **Cooperative Human-Machine System** can be defined as a set of agents, either human (2a) or machine (2b) (see Figure 9) for the items associated with these numbers). The agents interact and communicate (3). Tasks (4) are assigned to the cooperative system and allocated (6) to the agents who achieve them thanks to some resources (5). Each agent has access (7) to specific resources. The cooperative system operates on one or more controlled processes (8). It is, with the processes it controls, immersed in an environment (9) (e.g., weather, communication infrastructure). The internal (2a, 2b, 3, 4, 5, 6, 7) and external (8, 9) context of Cooperative Human-Machine Systems is inherently dynamic in many aspects: internally e.g. the tasks (4) are progressing, the capabilities of operators (2a) may degrade due to stress or fatigue, automated systems (2b) may fail or degrade; externally e.g. the environment (9) in terms of traffic and weather changes continuously.

In this deliverable we want to present how such distributed human-machine cooperative systems can be modeled.
We first have to explain how the notions of executive and adaptive functions fit within the distributed human-machine cooperative systems picture.

2.1.2.3.2 Distribution of executive and adaptation functions within the cooperative system

The notion of executive and adaptive control loops fits perfectly within the cooperative human-machine system that underlies HoliDes:

- some of the agents in the cooperative system close one or more executive loops on the external process/plant/vehicle
- some of the agents close one or more adaptive loops on these executive loops

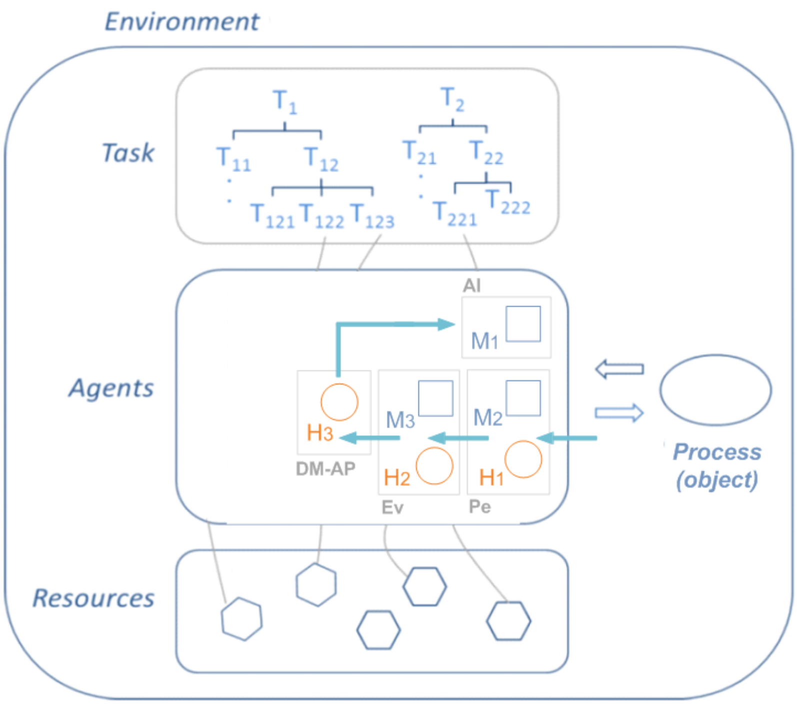

Figure 10: Executive loop in a CoS

In the Figure 10, a series of human and machine agents close an executive loop on a process P, to perform a series of tasks using some resources:

- H1 (human agent) and M2 (machine agent) *perceive (Pe)* the state of the process. Typically H1 is assisted by M2 (e.g., augmented reality).
- H2 and M3 *evaluate (Ev)* the information perceived by H1 and M2
- H3 gets the evaluations from H2 and M3 and performs *decision-making (DM)* to determine if an action on the process is needed, and which one.
- H3 then performs *action planning (AP)*, which consists of determining how to implement the foreseen action on P.
- a machine agent, M1, then *implements (AI)* the *action plan* specified by H3.

We see how this figure is an instantiation of the general schema for CoS (cooperative human-machine systems) in Figure 9, with many human and machine agents interacting and collaborating on performing the tasks on the process P, using specific resources. The difference is that we now see how the collaboration between the agents is organized around a single executive loop (on P).

The cooperative system above is a pure CoS: it does not exhibit any adaptive capabilities (i.e., it only executes an executive loop and that loop cannot be adapted, for example when the tasks to perform or the resources to use change, or if the environment in which the CoS behaves changes in a way that affects its performance).

We will now see how adaptive capabilities can be brought to the system, to get an AdCoS, a truly adaptive cooperative system. To have adaptive capabilities, the AdCoS must implement an adaptive loop that adapts the executive loop to the changing circumstances to which adaptation is required.

In Figure 11, an adaptive control loop is installed on the executive loop. Two agents, H4 (human) and M4 (machine) are added to the CoS. Both of them are exclusively in adaptive functions. H3, already in the executive loop, is also involved in the adaptive loop. H3 therefore superposes executive and adaptive functions.

The adaptive loop (on the executive loop) is closed in the following sequence: information on the executive loop (e.g. state of the agents, actions, performance...) is *perceived* by H3 and M4.

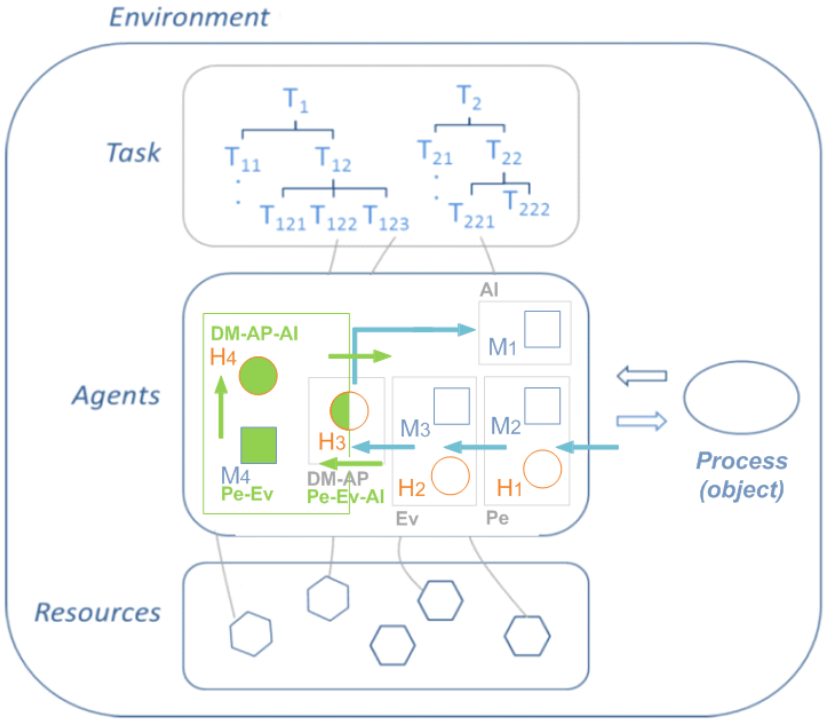

Figure 11: Executive and adaptive loops in an AdCoS

That information is then evaluated by H3 and M4 again, and made available to H4, the main adaptive agent, through direct Human to Human (H2H) communication from H3, and through a dedicated HMI (6) from M4. H4 performs *decision-making* and then *action planning*. The action *plan*, i.e., adaptation to the executive loop, is *implemented* by H4 itself and by H3.

The addition of an adaptive loop to the executive loop of Figure 10 transforms the CoS into an AdCoS.

2.1.2.3.3 Interactions between agents: H2H, H2M and M2M interactions

Interaction between the agents in the AdCoS, for coordinated performance of the tasks, synchronization and information passing, is achieved through specific Human to Human (H2H), Human to Machine (H2M) and Machine to Machine (M2M) modalities.

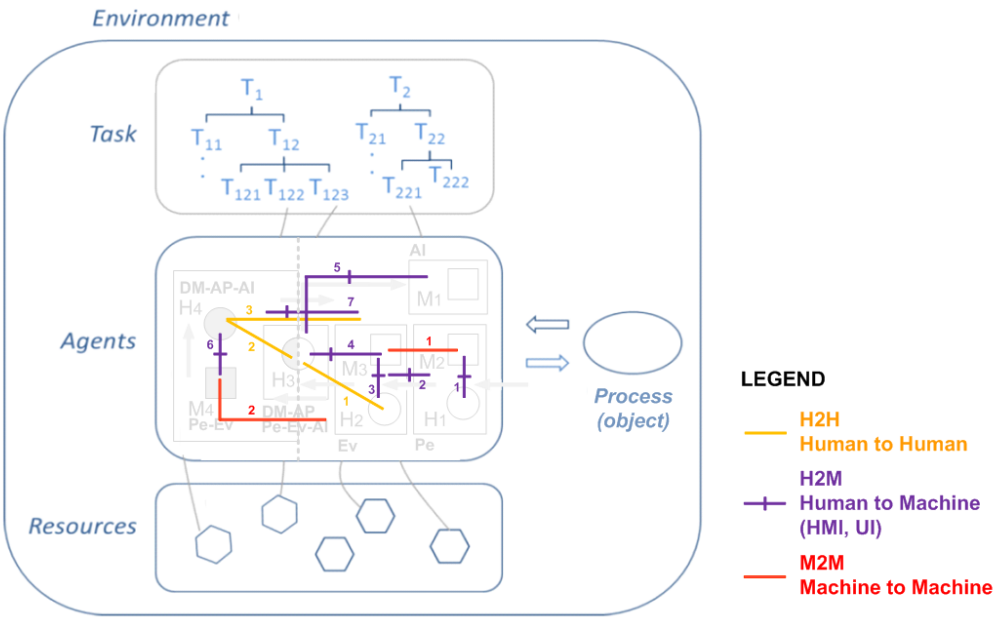

Figure 12: Inter-agents interactions within the AdCoS (H2H, H2M and M2M)

In Figure 12, the modalities and interaction types involved in the example AdCoS are shown.

For the "CoS" part (CoS = **Co**operative **S**ystem):

- Interaction between H1 and M2 is supported by H2M (human to machine) communication, that is some HMI or UI (**H2M 1**).
- The information perceived by H1 and M2 is made available to H2 and M3 through a dedicated HMI (**H2M 2**) (for H2) and M2M communication (**M2M 1**) (for M3). H2 and M3 also interact through another dedicated HMI (**H2M 3**).
- H3 gets the evaluations from H2 and M3 through some HMI (**H2M 4**) (from M3) and direct H2H (human to human) communication (**H2H 1**).
- The action plan produced by H3 (after decision-making) is passed to M1, for plan implementation, via a dedicated HMI (**H2M 5**).

For the "Ad" part (Ad = **Ad**aptive):

- H3 perceives the information on the executive loop directly (given that agent is already part of that loop).

- M4 perceives information on the executive loop through M2M communication (**M2M 2**), with the other machine agents in the executive loop (M1, M2 and M3), and possibly through additional sensors.
- The evaluations performed by H3 and M4 on the executive loops are passed to H4 through direct H2H communication from H3 (**H2H 2**), and through a dedicated HMI (**H2M 6**) from M4.
- The action plan (adaptation plan) produced by H4 is sent through direct H2H communication (**H2H 3**) to the executive loop human agents (H1, H2 and H3) and through a dedicated HMI (**H2M 7**) for H2M communication to the machine agents (M1, M2 and M3).

2.1.2.3.4 Superposition of functions in agents

Generally, any agent in the system, human or machine, can superpose executive and adaptive functions, related to different control loops. In Figure 11, H3 superposes executive and adaptive functions. All other agents are exclusively in executive or adaptive functions.

2.1.2.3.5 Task distribution and resource allocation

In an AdCoS the distribution of tasks and allocation of resources to the agents is rarely static. While an adaptive loop can adapt any of the characteristic parameters of an executive loop (e.g., which agents are involved in the loop or how user interfaces are used for human-machine interaction), the allocation of the tasks to the executive loop and its agents is a frequent object of adaptation. The three examples above (Figure 6, Figure 7 and Figure 8) involve an adaptive agent (the pilot in Figure 6 and Figure 7; the pilot and the autoflight system in Figure 8) specifying a flight plan (the main input or task) to be processed by the executive loop (whose role to perform the task).

Adaptive loops typically allocate tasks to executive loops. And within an executive loop, the adaptive loop typically allocates sub-tasks to the agents involved in the different executive steps. For example, if each step is performed by a pair of human and machine agents (e.g., human agents assisted by machine agents), the adaptive loop can define the level of assistance - or task sharing - between the human and machine agents. In such a case, task distribution is adaptive. See Figure 13.

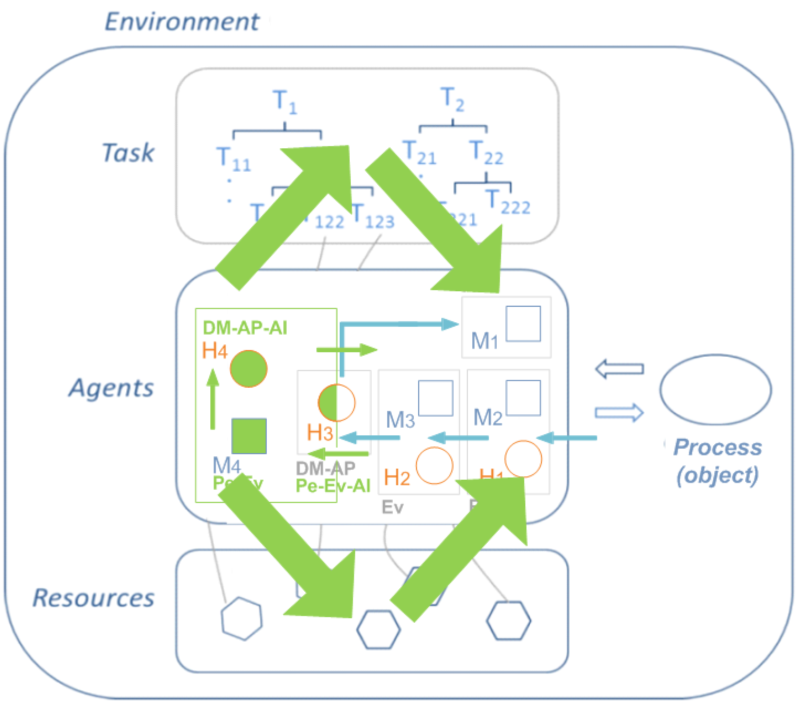

Figure 13: Adaptive task distribution and resource allocation

In the example of Figure 14 (from *D3.3 - Framework for Adaptation*), we see for example how task sharing between human and machine agents in charge of the steps in an executive loop are adapted between two different task distributions. That type of adaptation is performed via an adaptive loop that monitors various conditions, internal and external to the AdCoS, and decides what is the most appropriate task distribution/sharing at any time.

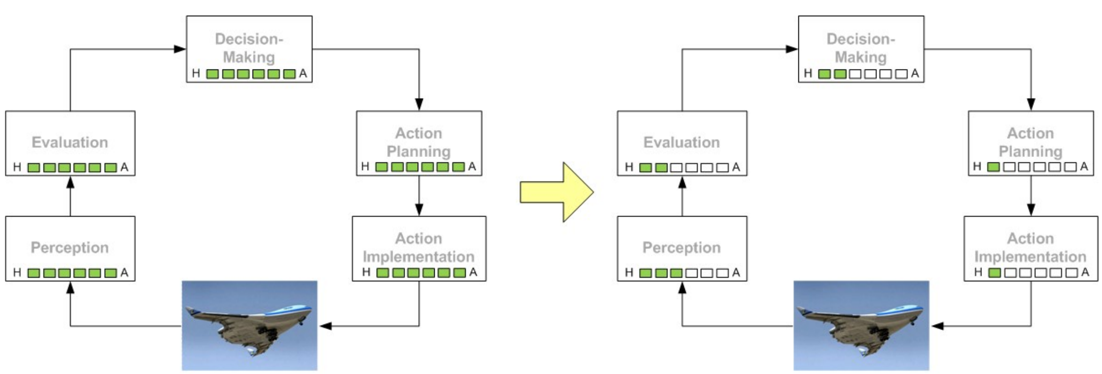

Figure 14: Adaptive task distribution

The same type of adaptive control can be applied to the resources made available to the agents for the execution of their tasks. In that case, resource allocation is adaptive.

2.1.2.3.6 Generalization

One will notice in the introduction on executive and adaptive functions in Section 2.1.2.2 and then in the figures above involving executive and adaptive loops that the two types of loops do not differ in nature: both types are control loops. Executive loops are loops on some external process (the "CoS" flavor). Adaptive loops are loops on executive loops (the "Ad" flavor).

In real life, most complex human-cooperative systems do not singlehandedly exhibit a single executive loop and a single adaptive loop (as in Figure 11). They may have several executive loops, several adaptive loops (on various executive loops), and even have control loops on adaptive loops, thus taking the shape of a hierarchy of control loops (many human organizations, inherently cooperative systems, take that shape). The end layer (leaves) performs the executive functions on the external processes controlled by the AdCoS. The upper layers perform the adaptation of the functions below them to various, internal and external, circumstances, a typical case being the adaptive distribution of tasks and allocation of resources (see Section 2.1.2.3.5) (as is indeed the case in many hierarchical human organizations).

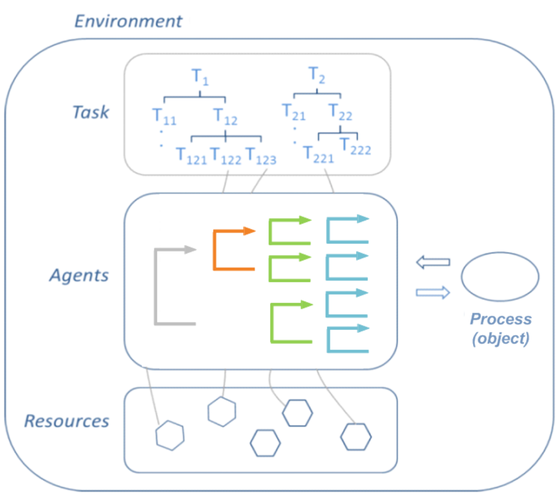

Figure 15: (Tree-like) Control structure in an AdCoS

The general case for an AdCoS is therefore one of **many intricate and inter-related control loops**, which can be analyzed in terms of a peculiar control structure, taking the shape of a **control tree** (e.g., in Figure 15: (Tree-like) Control structure in an AdCoS), and more generally of a **control graph** (normally acyclic, thus a Directed Acyclic Graph, DAG).

2.1.2.3.7 Implication for AdCoS modeling and design

AdCoS modelling consists in capturing all characteristics needed to understand the behavior of the AdCoS, in functional terms. AdCoS design consists in selecting and designing these characteristics:

- tasks
- resources
- process(es)
- environment(s)
- control loops: executive and adaptive loops, and more generally the associated tree- or graph-based control structure
- agents in the AdCoS, and their participation (static or dynamic) to the control loops, as well as the functions, services, algorithms etc. through which they perform their tasks
- task allocation
- resource allocation
- cooperation structure between agents

We will see in Section 2.2 how these characteristics can be modeled. In Section 2.3, we will see how they can be integrated with the HF-RTP, in the framework of a proposed AdCoS modelling methodology.

2.1.3 Graphical formalism

In *D3.4-V01-Techniques and Tools for Adaptation* we have introduced a graphical formalism for representing adaptive human-machine cooperative systems (AdCoS). The formalism is based on the dissociation between executive and adaptive functions. An example of an AdCoS characterization with that formalism is shown in Figure 16.

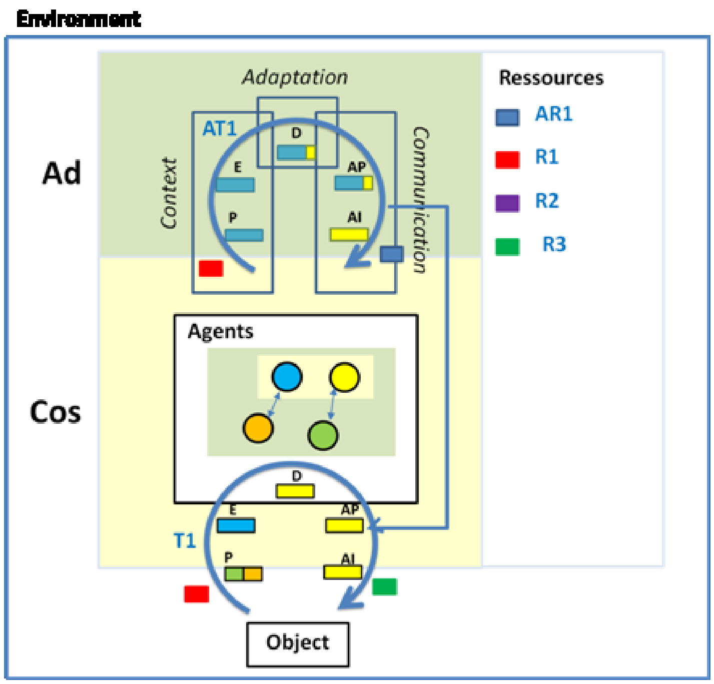

Figure 16: Graphical formalism for depicting adaptive control loops in AdCoS

The graphical formalism is intended to captures the main features of adaptive situations, like the agents involved, the operations shared by human and machine agents (i.e., the operations where machine agents provide assistance to human agents), the objects of adaptation (e.g., task sharing between agents), and the means used for that purpose. It cannot be considered a fully featured modeling formalism adequate for HoliDes objectives. In this deliverable, and later in HoliDes, we will therefore aim at producing more detailed and better specified AdCoS models based on the same ideas (executive and adaptive loops), define some methodologies for using them, and integrate the whole modeling and design framework into the HoliDes HF-RTP.

2.2 Model to build the Adaptation Framework

In this section, we analyze the AdCoS features that need to be modeled, in particular to support a deep understanding of AdCoSes and later to support a fully model-based AdCoS design methodology. We then investigate these features, how they need to be modeled and suggest general means of modelling them.

2.2.1 What needs to be modeled

We list here the AdCoS features that need to be modeled.

Executive loops

For each executive loop	
Object	Object or process on which the executive loop is closed. The loop is a control loop and the objective of the loop is to control the object or process. For example, for the MRI practitioner, the object of the executive loop is the patient, or at least the position of the patient in the MRI machine.
Environment	Environment in which the loop is executed. The environment includes all factors that impact: - the object of the loop - the agents - the tasks - the resources For example, for the MRI practitioner, the environment is the room where the MRI machine and the patient are placed, and more largely the hospital in which the operation is performed.
Agents	Executive agent(s) involved in the execution of the loop (human and machine agents). These agents - human and machine - define the composition of the human-machine cooperative system in charge of executing the executive loop. For example, in an aircraft, for an executive loop in charge of controlling the aircraft trajectory, the executive agents are the PF (pilot flying), PNF (pilot non flying) and the autoflight system (notably autopilot).
Tasks	Tasks involved in the execution of the loop.

	In the typical task taxonomy used in HoliDes (e.g., see *D1.4, D3.2*), the tasks in an executive loop are organized within 5 categories: perception, evaluation, decision-making, action planning, action implementation. For example, in the Diversion Assistant use case, five tasks have to be executed: perceive distance to current destination airport, evaluate the distance to see if it's acceptable, decide if the distance is not acceptable between possible diversion airports and select one, define an action plan to change destination to that new diversion airport, and implement this action plan.
Resources	Resources involved in the execution of the loop. These are all the resources used by the agents for performing their tasks. Given there may be different task categories, it is typically possible to organize the resources accordingly (e.g., perceptive resources, evaluative resources, action implementation resources). For example, sensors are perceptive resources and effectors are action implementation resources.
Task allocation	Allocation of tasks to the executive agents. The allocation of tasks to the agents is defined by a mapping: - task ⇨ agent(s) Correspondingly this allows determining all tasks currently assigned to a given agent: - agent ⇨ tasks Allocation of tasks is where the respective allocation of tasks to the human and machine agents is described. This therefore implicitly covers notions such as: - task sharing - assistance levels - automation levels

	See *D3.2* for examples of task sharing, corresponding to different assistance and automation levels.
Resource allocation	Allocation of resources to the executive agents. The allocation of resources to the agents is defined by a mapping: - resource ⇨ agent(s) Correspondingly this allows determining all resources currently assigned to a given agent: - agent ⇨ resources For example, to control an aircraft trajectory, some of the resources used by the PF, PNF and autoflight systems (executive agents) are various sensors (e.g., position sensors, aircraft attitude sensors, speed sensors,...) and effectors (e.g., the control surfaces,...).
Algorithms	Algorithms executed by the executive agents (human and machine) for performing their tasks. For performing a given task, an agent executes a particular algorithm. In the Diversion Assistant example above, specific algorithms are for example used by the machine agents to perceive the distance, evaluate the appropriateness of that distance (based on complex factors such as weather, fuel remaining, etc.) and then propose alternative diversion airports to the crew.
Cooperation structure	This is about the cooperation structure between the executive agents: some executive agents cooperate with others in the performance of some joint task (typically an executive step). The task is dissociated into subtasks, which are then distributed or shared between the agents. The cooperation structure captures which agent cooperates with which agents. It can be implicitly stated, or implicit, as a byproduct of task definition

	and task allocation.

Table 1: Executive loops description

One will note two important points:

- All the elements above in a given AdCoS can be dynamic, i.e., they change over time during AdCoS operations. For example new agents can come in or activate, some agents may leave the AdCoS or deactivate, new tasks may be assigned to the AdCoS, the resources may evolve, in particular as an effect of resource consumption, etc.
- All the elements above will be a possible object of adaptation (the "what"). Thus an adaptive loop is a control loop on one or more of these elements. This is by the way one of the ways in which dynamicity of the elements (cf. above) is brought into the AdCoS.

Adaptive loops

For each adaptive loop	
Objects	The executive loop(s) on which the adaptive loop is closed (an adaptive loop is indeed closed on one or more executive loops). For each of these executive loop(s): - The elements of the loop that are the object of adaptation. Therefore any of the items in the table above characterizing executive loops. The executive loops - and their specific elements - on which the adaptive loop is closed correspond to the "What" of the adaptation. This is of course a key component of an AdCoS description. For example, in a setting where a human agent is closing an executive loop with the assistance of several machine agents (e.g. car driving, with various assistance system), at any given time, there is a specific task sharing between the human agent (driver) and the machine agents (assistance

	systems). An adaptive loop is then closed over that executive loop to adapt the task distribution (or sharing) between the human and machine agents. That loop can be closed by a machine agent that observes the human agent, his/her vigilance and distraction state, and adapt task assistance accordingly, with total and temporary removal of the human agent of some of the critical steps (in the executive loop) if needed.
Environment(s)	The environments in which the adaptive loop is closed. Typically the environments in which an adaptive loop is operating are obtained by performing the union of all controlled executive loops' environments, which are inherited by the adaptive loop controlling them. For an adaptation loop on the MRI machine, the environment is the same as for the executive loop: the room in which the MRI operation is performed and the hospital. In the car driving example above, the machine agent doing the adaptation of course operates in the same environment then the executive agents (that is the car, the road, weather, traffic, etc.).
Agents	The adaptation agent(s), involved in the execution of the adaptive loop (human and machine agents). As noted above (see Section 2.1.2.2.3), in many cases, the agents are capable of self-adaptation and therefore superpose executive and adaptive functions. In this case, the executive and adaptive agent is the same (A=B). Thus in many cases, in more complex settings with multiple human and machine agents in the AdCoS, there is a significant intersection of the executive agents and the adaptive agents. When this intersection is empty, we are in a pure master-slave relationship, where the behavior of the executive agents is under complete control of the adaptive agents. In the car driving example, we have a single -

	machine - agent in charge of adapting task sharing between the human driver and the assistant systems in the executive loop.
Tasks	The adaptive tasks involved in the execution of the adaptive loop. Again, typically, these tasks can be organized into several categories, such as perception, evaluation, decision-making, action planning and action implementation; or context assessment, adaptation (computation) and communication. In the car driving example, the machine agent in charge of the adaptation will need to perceive and evaluate the state of the driver, decide if that state is appropriate or not (e.g., fatigued, distracted), and possibly decide to adapt task sharing in the executive loop (i.e., the level of assistance provided to the driver by the assistant systems). It will then define how to communicate appropriate orders (to the assistant systems) and information (to the human driver) and implement these actions.
Resources	The resources involved in the execution of the adaptive loop. These are the resources used by the adaptive agents to perform their tasks on the target executive loops. The resources are therefore used to take information on all relevant elements of the executive loops (cf. executive loops table above), such as on the state of the object, environment, agents, their tasks and resources, etc. and affecting them (e.g., communication means used by the adaptive agents to communicate the adaptations to the executive agents). In the car driving example, the main resources used by the adaptive machine agent are various types of sensors and calculators involved in perceiving the state of the driving.
Task allocation	The allocation of adaptive tasks to the adaptation agents.

	The allocation of tasks to the adaptive agents is defined by a mapping: - tasks ⇨ agent(s) Correspondingly this allows determining all tasks currently assigned to a given adaptive agent: - agent ⇨ tasks In the car driving example, we only have a single machine agent in charge of closing the adaptive loop. Thus there is no dynamic task allocation: that agent is active at all time and doing the same thing, closing all loop steps permanently.
Resource allocation	The allocation of resources to the adaptation agents The allocation of resources to the adaptive agents is defined by a mapping: - resource ⇨ agent(s) Correspondingly this allows determining all resources currently assigned to a given adaptive agent: - agent ⇨ resources In the car driving example, the machine agent in charge of closing the adaptive loop will typically use the same resources (for driver state perception) at all time. These resources will not vary or be adapted. There is therefore no (dynamic) resource allocation in this example.
Algorithms	The algorithms executed by the adaptive agents (human and machine) for performing their tasks. The algorithms used by the agents correspond to the "How" of the adaptation. In the car driving example, the machine agent in charge of the adaptive loop will need various algorithms, for the different steps it has to go through when closing the adaptive loops: algorithms for perceiving the state of the driver (including merging and consolidating data from various types of sensors); for evaluating the appropriateness of the state of the driver (in relation to the current context);

	then critically to decide if task sharing (in the executive loop) has to be changed; and then to alter task sharing accordingly and communicating with the driver.
Cooperation structure	The cooperation structure between the adaptive agents (see cooperation structure for executive agents above for details). In the car driving example, given there is a single adaptive agent - the machine agent in charge of adapting task sharing in the executive loop - there is no cooperation structure. The agent is alone in charge of adaptation.

Table 2: AdCos features of an Adaptive Loop

2.2.2 Types of Models

In this section, we are looking at potential models and formalisms for modelling the different types of elements involved in executive and adaptive loops. For each type, we first specify what has to be modeled, and any peculiarity of interest, and then list corresponding modelling frameworks.

Object/Process	
An object on which an executive loop is closed should be seen as a dynamic process. Which means that it has states and events, and typically includes its own dynamicity (i.e., its states change without any action from the executive loop).	
Models	
State machines	https://en.wikipedia.org/wiki/Finite-state_machine
Statecharts	https://en.wikipedia.org/wiki/State_diagram
Petri Nets	https://en.wikipedia.org/wiki/Petri_net
Hybrid models	Models that combine discrete and continuous features. https://en.wikipedia.org/wiki/Hybrid_system

Environment
An environment, in which executive and adaptive loops are immersed, should also be seen as a dynamic process. An environment has its own dynamicity and normally cannot be influenced: it has states and events that are not influenced by the executive and adaptive loops (otherwise it would be an object, i.e., something that is impacted and/or controlled by the loops).

The models usable to characterize environments are therefore the same than for objects.	
Models	
State machines	https://en.wikipedia.org/wiki/Finite-state_machine
Statecharts	https://en.wikipedia.org/wiki/State_diagram
Petri Nets	https://en.wikipedia.org/wiki/Petri_net
Hybrid models	Models that combine discrete and continuous features. https://en.wikipedia.org/wiki/Hybrid_system

Agents	
An agent, human or machine, is also a dynamic process. Technically, there are two ways of seeing agents: - *intentional agents*: Intentional agents have goals they are trying to achieve. Their behavior can be understood in terms of these goals and of the mechanisms put in play to achieve them. The most appropriate approach within HoliDes to characterize these agents is to model their goals in terms of sub-goals and associated tasks, and their mechanisms. - *non intentional agents*: Non intentional agents do not have goals. They are seen as a pure dynamic process, and can therefore be modeled with the same type of tools as objects or environments. They behave mechanically, typically in a reactive way [e.g., event ⇨ action; event, state ⇨ action]. The most appropriate approach to characterize these agents is therefore in terms of correspondence tables or state machines (or analog). The intentional perspective is especially useful for purposeful and complex agents. The non intentional perspective is more useful for non purposeful, mechanistic and simple agents.	
Models	
Intentional agents	
Cognitive architectures	Cognitive architectures are general architectures for implementing intentional agents. Cognitive architectures aim at capturing and simulating human cognitive behavior. For a good overview of cognitive architectures, see https://en.wikipedia.org/wiki/Cognitive_architecture including with a rather exhaustive list of existing

	architectures. Of particular interest in HoliDes is CASCaS [13], the cognitive architecture developed by OFFIS which can be programmed with sets of rules specifying the behavior of the cognitive agent. The architecture then interprets the rules to generate dynamic behaviors. Also, see *Deliverable D1.4 - HF-RTP-1 0*, Section 2.7.2. Other famous general cognitive architectures include SOAR [14], ACT-R [15]and ICARUS[16].
Non intentional agents	
State machines	https://en.wikipedia.org/wiki/Finite-state_machine
Statecharts	https://en.wikipedia.org/wiki/State_diagram
Petri Nets	https://en.wikipedia.org/wiki/Petri_net
Hybrid models	Models that combine discrete and continuous features. https://en.wikipedia.org/wiki/Hybrid_system

Tasks	
Tasks incorporate two main dimensions: - *hierarchical dimension*: Tasks are made of subtasks, recursively, until some primitive, terminal low level tasks are reached. That aspect can be modeled through hierarchical structures, typically trees. - *temporal dimension*: Tasks exhibit temporal relationships, based on specific temporal orders. That aspect can be modeled with graphs. HoliDes would benefit in using task modeling formalisms that incorporate both aspects.	
Models	
PED	The PED task editor, developed by OFF, see *Deliverable D1.4 - HF-RTP-1 0*, Section 2.7.2.
UML activity diagrams	https://en.wikipedia.org/wiki/Activity_diagram
UML state charts	https://en.wikipedia.org/wiki/State_diagram_(UML)
CTT	https://en.wikipedia.org/wiki/ConcurTaskTrees

Resources
A resource in the HoliDeS AdCoS framework is anything that is used by the AdCoS agents to achieve their tasks. As for tasks, resources can be allocated to - or used by - several agents at the same time. Complex schemes then arise, depending on the resource properties. For

example:

- *consumable resources*: They are depleted when used (e.g., fuel in a car or an aircraft). Their capability to be used for task achievement decreases or even stops abruptly. Many depletion schemes are possible.
- *non consumable resources*: They are not depleted when used and therefore provide constant capabilities for task achievement when they are available (e.g., a user interface used to gain information on patient positioning in the MRI machine or the gear lever in an aircraft)
- *reloadable resources*: consumable resources that can be reloaded, to restore capabilities that had been lost, due to resource depletion
- *non reloadable resources*: the opposite. Cannot be reloaded and therefore lost if depleted.
- *availability*: availability or non-availability of the resource. Some resources are only available in specific conditions (e.g., sunlight is only available during the day)
- *shareability*: capacity of the resource to be used by several agents at the same time. Resources subject to mutual exclusion can only be used by a single agent at a time (e.g., a runway for an aircraft). In some cases, the resource is shareable, but by a limited number of agents at a time (e.g., a waiting zone before a runway that has a limited capacity).

These are just examples and resources in general have many different properties that determine how they are used and shared by the agents in an AdCoS. This makes their modelling difficult and dependent on the properties to be modeled.

Models	
State machines	https://en.wikipedia.org/wiki/Finite-state_machine
Statecharts	https://en.wikipedia.org/wiki/State_diagram
Petri Nets	https://en.wikipedia.org/wiki/Petri_net
Hybrid models	Models that combine discrete and continuous features. https://en.wikipedia.org/wiki/Hybrid_system

Task allocation

Task allocation in a CoS can take two forms:

- task ⇨ agents: A task is allocated to one or more agents.

- agent ⇨ tasks: An agent is in charge of one or more tasks.

In both cases, task allocation takes the form of a correspondence relation, and can therefore be modeled through simple correspondence tables.

Models	
Correspondence table	Anything allowing specifying such tables. For example Excel.

Resource allocation

Resource allocation, as for task allocation, can take two forms:
- resource ⇨ agents: A task is allocated to (or used by) one or more agents.
- agent ⇨ resources: An agent uses one or more resources.

In both cases, resources allocation is seen as a correspondence relation and can be modeled through simple correspondence tables.

Models	
Correspondence table	Anything allowing specifying such tables. For example Excel.

Algorithms

An algorithm is a method used for performing specific operations or computations. According to Wikipedia: "An algorithm is an effective method that can be expressed within a finite amount of space and time and in a well-defined formal language for calculating a function". In HoliDes, algorithms are needed to specify how the agents involved in a control loop (executive or adaptive) perform the tasks assigned to them (e.g., an algorithm for extracting information from the environment and presenting that information to a human user for further evaluation). The description, specification and execution of algorithms is the subject of computer science and can take many forms. From Wikipedia again: "Algorithms can be expressed in many kinds of notation, including natural languages, pseudocode, flowcharts, drakon-charts, programming languages or control tables (processed by interpreters)".

Models	
Pseudo-code	https://en.wikipedia.org/wiki/Pseudocode

Flowcharts	https://en.wikipedia.org/wiki/Flowchart
Programming languages	https://en.wikipedia.org/wiki/Programming_language
Control tables	https://en.wikipedia.org/wiki/Control_table

Cooperation structure

The cooperation structure is therefore about cooperative relations that exist between agents in the AdCoS. Agents cooperate when they perform sub-tasks that contribute to a common super-ordinate task (i.e., the task is split into smaller sub-tasks and the sub-tasks are shared between the agents).

As mentioned above, the tasks and their allocation to the agents imply a corresponding cooperation structure: The cooperation structure is implicit in the definition of the tasks and their allocation.

It can however be interesting to directly reason on the cooperation structure (in particular in terms of task sharing) and derive at later times the corresponding tasks definitions and allocation.

Basically, the cooperation structure can be defined as a set of items of the type [T, [(A1, t1), (A2, t2)... (An, tn)]] where:

- T is a super-ordinate task
- A1, A2,... are the agents involved in the super-ordinate task
- t1, t2... are the sub-tasks assigned to the agents. The composition of all sub-tasks is normally equivalent to the super-ordinate task T.

Each such item defines a cooperative unit, in charge of a given super-ordinate task. At any given time, there may be several such units active in an AdCoS. Given a sub-task can be the super-ordinate task of another unit, the AdCoS cooperative structure takes the shape of a graph (with a tree-like structure for example if the allocation of tasks to the agents in based on hierarchical task decomposition).

Models	
Tables	A table listing the cooperative units in the AdCoS
Graph	A graph linking the cooperative units in a super-ordinate/sub-tasks relation.

2.3 Integration of the Adaptation Framework in the HF-RTP

The different object types, and their modelling, must be integrated in the HF-RTP. The HF-RTP provides an integrated set of methods, techniques and tools for understanding, modelling and designing AdCoS, with major considerations paid to Human Factors aspects.

2.3.1 The HF-RTP

The *Deliverable D1.4* provides a description of the possible contributions of the HF-RTP to the Adaptation Framework and AdCoS modelling. Section 2.7 provides a series of general principles on how to achieve interoperability, at the semantic and technical levels. It also lists standards the Adaptation Framework and AdCoS modelling methodology should try to comply to. In particular:

- interoperability standards. See details in *Deliverable D1.4*
  - e.g., HTML
  - REST
  - CSW
  - DCAT
- tool protocols
  - HTTP & HTML
  - RTPS
  - TCP IP/UDP
- programming interfaces
  - cloud computing tools. See applicable standards in Deliverable D1.4, table 9
  - interface tools/frameworks
  - functional mock-up interfaces
    - DIPS
    - IBM Rational Jazz
    - Apache CXF
    - Open XC (automotive domain)
    - AUTOSAR (automotive domain)
- runtime aspects
- communication paradigms. Standards:
  - IEEE Standard for modelling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules.

- ETSI standards, mainly for Radio & Telecommunications Terminal Equipment

- exchange formats:
  - formats: RDF, XML, JSON and CVS
  - parsing & conversion tools

Deliverable D1.4 also list in Section 2.7.2 a series of tools, available outside of HoliDes or developed within HoliDes the modelling methodology should attempt to rely on (e.g., PED, for task modelling and specification; COSMOCivic, for driver modelling; CASCaS, cognitive architecture for general human agent modelling etc.).

2.3.2 An AdCoS modelling methodology

We propose below an overall methodology for AdCoS modelling in the Adaptation Framework from *Deliverable D3.3*, as explained and further expanded in the Section 2.1.

2.3.2.1 Working hypotheses

That methodology relies on the distributed cooperative system (DCoS) approach that underlies HoliDes, the notions of executive and adaptive control loops and the following working hypotheses:

- The AdCoS is seen as a system of human and machine agents that cooperate on the performance of some tasks on some object(s), through some resources. See Section 2.1.2.3.1 and in particular Figure 9.
- This is achieved by closing a series of executive loops on these object(s), see Section 2.1.2.2.
- The executive control loops - and how they are performed by the AdCoS agents - can be adapted through dedicated adaptive loops, providing an adaptive flavor ("Ad") to the cooperative system ("CoS"), see Section 2.1.2.2.
- The multiple executive and adaptive loops can be intertwined through complex loop control structures, see Section 2.1.2.3.6 and Figure 15: (Tree-like) Control structure in an AdCoS.

- The AdCoS agents themselves participate, in various task distribution or sharing and resource allocation schemes, in the performance of these loops,see Section 2.1.2.3.5.
- An AdCoS can be understood and modeled in terms of such a structure and its documentation. Symmetrically, an AdCoS can be designed, in a model-based approach, through the definition of such a control structure, its individual control loops, their steps, and the assignation of agents to these steps (task distribution), with the corresponding assignation of resources to exploit for performing the tasks (resource allocation).

2.3.2.2 Further integration with WP1 and the HF-RTP

The AdCoS modeling methodology is intended as a first step towards AdCoS design. The Adaptation Framework (*Deliverable D3.3*) and the modeling ideas presented here to capture and formalize that framework will be fully integrated with the activities in WP1, and in particular in the HF-RTP. That work will therefore be further continued in HoliDes.

2.3.2.3 Steps of the modeling methodology

The AdCoS modeling methodology proceeds through the following steps:

Determination of executive and adaptive loops in the AdCoS (Section 2.3.3) • Determination of executive loops (Section 2.3.3.1) • Determination of adaptive loops (Section 2.3.3.2) • Modelling of loops control structure (Section 2.3.3.3)

Modeling of executive and adaptive loops (Section 2.3.3.3)
• Selection of the loops to characterize and model (Section 2.3.4.1)
• Characterization and modelling of the loops (Section 2.3.4.2)
    ○ steps (Section 2.3.4.2.3)
    ○ tasks (Section 2.3.4.2.4)
    ○ resources (Section 2.3.4.2.5)
    ○ agents (Section 2.3.4.2.6)
    ○ task distribution (Section 2.3.4.2.7)

○ resource allocation (Section 2.3.4.2.8) ○ interactions between agents (M2M, H2M, M2M) (Section 2.3.4.2.9) ○ cooperation structure between agents (Section 2.3.4.2.10)

Table 3: Methodological steps for model-based AdCos modeling

2.3.2.4 HF-RTP techniques and tools

In the following sections, we will try to find HF-RTP techniques and tools appropriate for the different modeling steps. The tools available so far in the HF-RTP are described in Section 2.7.2 of *Deliverable D1.4 - HF-RTP 1.0*.

For some steps, no specific tools are provided by the HF-RTP and this will impact the continuation of this modeling work: Finding appropriate tools that can serve these needs and be easily integrated in the HF-RTP (also confer Section 2.3.2.2 on further integration of the modelling methodology in the HF-RTP). In many cases for example, UML (2.4 or 2.5) can be used to cover all needs. For that reason, the sections below suggest the types of UML diagrams appropriate for each step.

Besides dedicated tools applicable to specific steps, the HF-RTP provides more general and integrated tools, mostly software frameworks, analysis suites, or simulation frameworks that allow simulating the totality - or at least significant parts - of an AdCoS, the object(s) it is operating on and the environment(s) in which is it immersed. Because these different types of tools are transverse to the methodology and/or integrative in nature we present them here.

Integrated and transverse HF-RTP techniques & tools	
SearchBestie	Software analysis (BUT)
Race Detector & Healer for Java	Software analysis (BUT)
AnaConDA	Software analysis (BUT)
Predator	Software analysis (BUT)
ProSIVIC	Simulator (CIVITEC)
RTMaps	Software framework (INTEMPORA)
MOVIDA	Simulator (IFFSTAR)
ADAS	Simulator (IFFSTAR)

Table 4: Integrated and transverse HF-RTP techniques & tools

We only present here the tools usable for *modeling*. Other tools are provided in the HF-RTP (such as the Human Efficiency Evaluator by OFFIS or the AudioDistraction algorithms and tools by TWT), but given they are more useful for AdCoS *design* than AdCoS modeling they are not presented here. *See Deliverable D1.4 - HF-RTP-1 0*, Section 2.7.2.

2.3.2.5 Illustration of the methodology through an example

To illustrate the methodology and its different steps, we will apply them on a simple but nonetheless important and realistic example: **the cockpit of a commercial airliner**, seen as an adaptive cooperative human-machine system, which is an AdCoS. We will analyze the AdCoS associated with the control of the aircraft (A/C) trajectory (there are indeed other AdCoS in a cockpit).

The AdCoS for the control of the A/C trajectory has five agents.

- human agents:
  - Pilot Flying (PF)
  - Pilot Non Flying (PNF)
- machine agents:
  - Autopilot (AP)
  - Auto-throttle (A/THR)
  - Flight Management System (FMS)

Figure 17 shows the AdCoS and (a subset of) what has to be modeled.

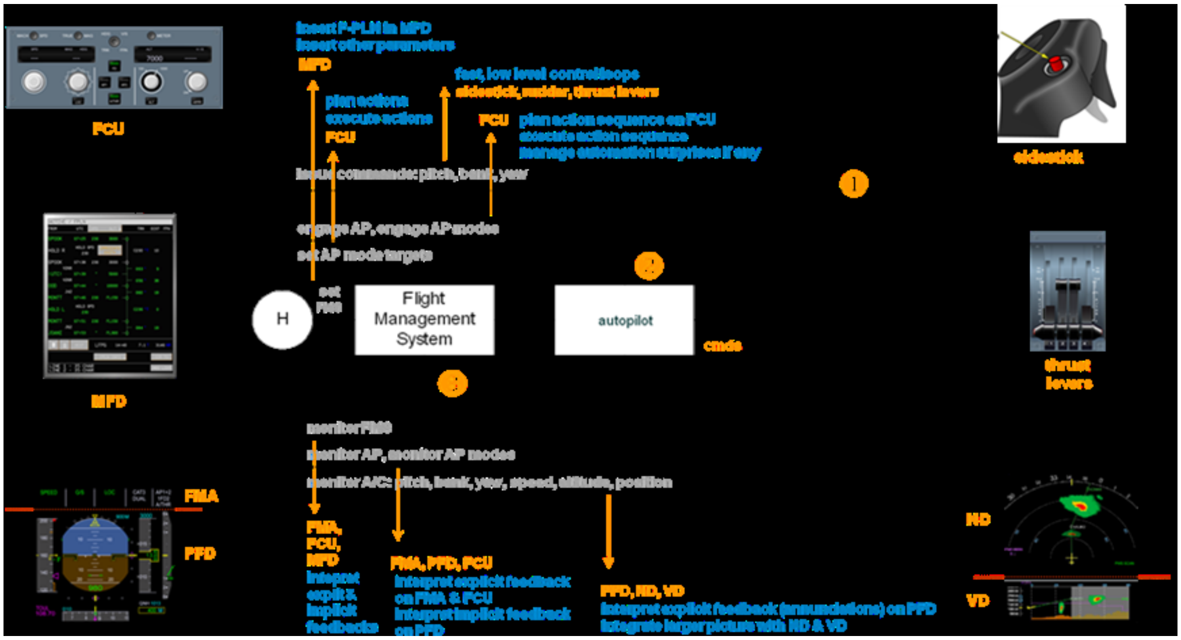

Figure 17: Commercial Aircraft (A/C) cockpit (A380) example

The figure shows:

- the *object*: The aircraft (A/C) is the object/process on which an executive loop is closed.
- the three *AdCoS loops*: an executive loop on the A/C, an adaptive loop on the autopilot and an adaptive loop on the FMS
- the *agents*: H (the crew, of Pilot Flying (PF) and Pilot Non Flying (PNF)), the Flight Management System (FMS) and the autopilot. H is made of two human agents. The FMS and the autopilot are machine agents.
- *interactions between the agents*, and the H2M User Interfaces (UI) used for interaction between the human agents and the machine agents
- some of the *tasks* the human which agents have to achieve in the completion of the different loops. The human agents are involved in all three loops

2.3.3 Determination of executive and adaptive loops in the AdCoS

This consists in determining all executive and adaptive loops in the AdCoS. The loops - executive and adaptive - are the basic unit of the analysis - and later of the design - of the AdCoS. The goal is to understand the AdCoS as a cooperative system of human and machine agents acting together to perform some control loops:

- control loops on some external object(s) the AdCoS is acting upon: executive loops
- control loops on the executive loops themselves: adaptive loops.
- control loops applying on the adaptive loops themselves (complex loop control structure, see Figure 15)

2.3.3.1 Determination of executive loops

Executive loops are concerned with what the AdCoS "does". To identify the executive loops in an AdCoS, one therefore has to look for the objects or processes on which the AdCoS applies. Once the objects are known, it's generally easy to derive the associated control loops, included the executive agents involved. For example, aircraft trajectory (object) controlled by crew and autoflight system (executive agent); patient positioning in MRI machine (object) controlled by practitioner (adaptive agent).

Applicable HF-RTP techniques & tools
Seemingly none so far.

<u>For the commercial cockpit AdCoS example (Figure 17):</u>

- The process under control is the A/C, and more exactly the A/C trajectory.
- There are four executive agents that can close the executive loop on that process:
  1. the PF (manual flight; or mixed flight, with the AP/A/THR providing guidance on the Flight Director (FD))
  2. the PNF (same. manual flight; or mixed flight, with the AP/A/THR providing guidance on the Flight Director (FD))
  3. AP (AutoPilot). More or less computes and/or control A/C trajectory
  4. A/THR (AutoThrottle). More or less computes and/or control engine thrust.

- Given it's possible to control the A/C trajectory in mixed mode (AP/A/THR providing guidance on the FD and then the crew executing the corresponding actions on the controls), the loop presents a possibility of task sharing.
- The environment is made of the geographical environment (e.g., terrain, mountains, elevations...), the meteorological environment, the radio-navigation installations on ground (e.g., VOR, ILSetc.) and the ATC.

UML modeling: UML modeling is not needed. The output of this step is **a list** of executive loops in the AdCoS.

2.3.3.2 Determination of adaptive loops

Adaptive loops are concerned with how what the AdCoS does is adapted. These are the aspects of the executive loops that are modified through adaptation. Executive loops can be characterized and specified on multiple aspects (see Section 2.2.1). Any of these aspects can be an object of adaptation. For example, the tasks assigned to the executive loop; the resources used; the agents, human or machine, performing the loop; task distribution or task sharing between these agents; etc. Thus, to identify the adaptive loops in an AdCoS, one has to look for what is adapted, amongst the many aspects of the AdCoS executive loops that can be adapted.

Applicable HF-RTP techniques & tools
Seemingly none so far.

For the commercial cockpit AdCoS example (Figure 17):
- The FMS closes an adaptive loop on the AP+A/THR, providing the guidance targets (tasks) to achieve. This is thus adaptive task allocation.
- The crew can also close the same loop, providing guidance targets (tasks) to AP+A/THR.
- The FMS and crew cannot cooperate within that loop, there is no possible task sharing like in the executive loop.

Secondary adaptive loop:

- There is a secondary adaptive loop, closed by the crew (PF and/or PNF), on the FMS.
- That loops exploits information on the state of the environment and of the A/C to determine if the current flight plan can be continued safely. If not, the plan (task) inserted in the FMS is adapted by the crew, after negotiation with the ATC.

UML modeling: UML modeling is not needed. The output of this step is **a list** of adaptive loops in the AdCoS.

2.3.3.3 Modeling of loops control structure

This more or less consists in determining which loop controls which loop(s), (see Section 2.1.2.3.6 and Figure 15).

In very simple AdCoS, there are few executive and adaptive loops, but in complex ones (e.g., the border control room use case), there are many. The relation between the loops, as well as what each loop controls particularly (e.g., task distribution or resource allocation between the agents within the controlled loop, see Figure 13), have to be described. Graphs such the one in Figure 15 can be used for that purpose.

Applicable HF-RTP techniques & tools
Seemingly none so far.

For the commercial cockpit AdCoS example (Figure 17):
- We have a linear control structure:
  - executive loop ⇦ adaptive loop 1 ⇦ adaptive loop 2

UML modeling: The output of this step is a graph-like structure interconnecting executive and adaptive loops. The graph must allow distinguishing between executive and adaptive loops (=type). The graph could also incorporate additional information such as the nature of the objects/parameters controlled for each loop. Besides that the graph should be kept simple. The goal is to visualize the overall AdCoS structure, seen as an organized set of loops. Details about the loops, the agents in them, etc. are provided in later models.

Tentative UML diagrams:
- component diagrams

2.3.4 Modeling of executive and adaptive loops

Each executive and adaptive loop in the AdCoS can then be characterized and modeled in detail.

Characterization means providing natural language descriptions or limited symbolic descriptions of some of the loop's features. Modeling means producing fully formal descriptions.

Depending on what really matters and what is really challenging in the understanding, and possible future design, of the AdCoS, some loops will only need characterization and others will require far more detailed modeling.

2.3.4.1 Selection of loops to characterize and model

Which loops to characterize and model is a matter of appreciation. At least, all adaptive loops need to be characterized and modeled. In particular, at a later design stage, algorithms for implementing those loops will need to be specified and implemented. It is also recommended to characterize and model all executive loops on which the adaptive loops are closed.

Applicable HF-RTP techniques & tools
Seemingly none so far.

For the commercial cockpit AdCoS example (Figure 17):
We will select *all three loops in the control structure* above. They are all important to understand the AdCoS.

UML modeling: UML modeling is not needed, given the output of this step is a list of executive and adaptive loops that have been selected for characterization and modeling.

2.3.4.2 Characterization and modelling of the loops

Once the executive and adaptive loops to characterize and model have been selected, they can be investigated and modeled, with a level of detail appropriate for the purpose. As stated above, depending on where the design challenges lie for the AdCoS, some loops will only need basic characterization, but others will need to be fully modeled. In particular modeling will be useful or needed if one intends to resort to model-based design.

2.3.4.2.1 Objects

For each **executive loop**, the object(s) (or processes) on which it applies has to be described.

Objects for adaptive loops do not need to be described here, given their objects, executive loops, will be described elsewhere. Thus, only the objects "external" to the AdCoS need to be described.

Applicable HF-RTP techniques & tools	
GreatSPN	University of Torino. See http://www.di.unito.it/~greatspn/index.html

For the commercial cockpit AdCoS example (Figure 17):
- *executive loop*: The object on which the executive loop is closed is the A/C, and in particular the A/C trajectory.

UML modeling: The output of this step is a set of objects on which the AdCoS executive loops are closed.

An object can be described in terms of:
- structure: i.e., the object's component and its interactions
- behavior: i.e., how the object, including its components, changes states.

In the proposed modeling approach, objects, environments and agents are all seen as instances of a more general class: processes.
- *objects*: processes **outside** the AdCoS, and controlled by the AdCoS

- *environments*: processes not under control of the AdCoS (They can only influence the AdCoS, not the reverse - otherwise they would be objects.)
- *agents*: processes, **within** the AdCoS, with tasks and therefore intentionality (goals).

The type of usable UML diagrams is therefore the same for the three types of processes.

For objects,

Tentative UML diagrams:
- structure: component diagrams
- structure: model diagrams
- behavior: (behavioral) state machine diagrams
- behavior: sequence diagrams
- classes: class diagrams (to organize objects into class hierarchies)

2.3.4.2.2 Environments

Environments are the environments in which the loops run. An environment is seen as a dynamic process that cannot be influenced by the loop (e.g., weather for an aircraft as the crew, in charge of controlling the aircraft, cannot influence the weather). If the loop could influence the environment, it would be an object.

Each loop may have one or more environments, based on the type and context of execution.

In many cases though, the environments for the AdCoS loops are all the same: They are the environments in which the AdCoS run. Each loop, being immersed with the AdCoS in these environments, inherits of them.

Applicable HF-RTP techniques & tools	
GreatSPN	University of Torino

For the commercial cockpit AdCoS example (Figure 17):

- *executive loop*: The environment is the environment in which the A/C flies, tentatively meteorological environment (weather).
- *adaptive loop1 and adaptive loop 2*: the cockpit

Thus there are two environments in the example: external to the A/C and internal to the A/C (cockpit). It is important to dissociate them. In particular the cockpit environment (e.g., fire, smoke etc.) can influence the two adaptive loops.

UML modeling: As explained above, environments in the proposed modeling resort from processes. Processes (objects, environments, agents) can all be described with the same UML diagrams.

For environments,

Tentative UML diagrams:
- structure: component diagrams
- structure: model diagrams
- behavior: (behavioral) state machine diagrams
- behavior: sequence diagrams
- classes: class diagrams (to organize environments into class hierarchies)

2.3.4.2.3 Steps characterization and modelling

Executive and adaptive loops can for example be dissociated into various steps. See for example the cognitive perspective on executive loops in Figure 2 (the same perspective can be used to analyze adaptive loops, see for example Figure 9). Given one of the objectives of HoliDes is to better introduce and support Human Factors (HF) during the design of cooperative human-machine systems, it is strongly recommended to characterize **each loop in which a human agent** is involved in those terms.

This means that performing the loop is seen as a cognitive problem, where the intervention of machine agents aside the human contributes to the cognitive performance of the loop. For loops in which human agents are not involved, other discretizations of the loops are possible (e.g., information taking, processing, action), though it is again strongly recommended to resort to a cognitive perspective if the machine agents in the loop act in a cognitive way. By resorting to the cognitive perspective to characterize each loop in the situation, one allows seeing the cooperative human-machine system as a distributed cognition system. Also, seeing - and designing - machine agents as cognitive agents also facilitates their interactions with human agents.

Applicable HF-RTP techniques & tools	
GreatSPN	University of Torino

For the commercial cockpit AdCoS example (Figure 17):

- **executive loop**
  The loop can be run in three (cooperation) modes:
  manual flight
  - perception:
    - PF, PNF (with assistance from other cockpit equipment)
  - evaluation
    - PF, PNF (with assistance from other cockpit equipment)
  - decision-making
    - PF, PNF
  - action planning
    - PF, PNF
  - action implementation
    - PF, PNF

  mixed flight
  - perception:
    - AP+A/THR
    - PF, PNF
  - evaluation
    - AP+A/THR
    - PF, PNF
  - decision-making
    - AP+A/THR
  - action planning
    - AP+A/THR
  - action implementation
    - PF, PNF

automatic flight
- all 5 steps performed by AP+A/THR.

- **adaptive loop 1**
  The loop can be performed in 2 modes:
  manual
  - PF, PNF perform all five steps of the control loop on the AP+A/THR
  automatic
  - the FMS performs all five steps of the control loop on the AP+A/THR

- **adaptive loop 2**
  The loop can be performed in only one mode.
  manual
  - PF, PNF perform all five steps of the control loop on the FMS.

UML modeling: The output of this step is, for each loop being modeled, a model that describes the agents that participate in the different loop' steps.

If the loop can be run in different cooperation modes (i.e., if there is adaptive task sharing, with various adaptive configurations of agents contributing to the loop), separate models have to be produced for each cooperation mode.

If the cooperation modes differ between steps (i.e., different agents participate in the steps, and/or within each step, task sharing differ), it will be better to describe each step separately, with its different local modes, and then the loop as a composition of steps. This can for example be documented with component UML diagrams.

Integration between the loops is normally not described here (though this could be done by interconnecting the component models for the different loops). This is done in the AdCoS loops structure modelling above, which provides a general architecture of the AdCoS in terms of loops, with a limited amount of detail, just to get a better general overview.

<u>Tentative UML diagrams</u>:
- component diagrams
- communication diagrams
- sequence diagrams
- activity diagrams

2.3.4.2.4 Tasks

These are the tasks performed by the AdCoS and distributed between its agents see Figure 9.

To identify the tasks, consider the executive and adaptive loops in the AdCoS, and more generally the whole of the AdCoS loops control structure, see Section 2.3.3.3.
- Determine the tasks performed by the loops.
- Determine how these tasks are split into sub-tasks assigned to the different task steps (e.g., perception, evaluation, decision-making ec.).
- If the steps are shared by several agents, further analyze the sub-tasks, until reaching the granularity level of the tasks assigned to the agents.

Applicable HF-RTP techniques & tools	
CTT	ConcurTaskTrees (CNR-ISTI). See http://www.w3.org/2012/02/ctt/
PED	Procedure Editor (OFFIS)

<u>For the commercial cockpit AdCoS example (Figure 17):</u>
The tasks can be easily derived from the loops decomposition (into five steps) (e.g., Figure 2). Also consider the "cockpit model" (Figure 17) for an example of a subset of the tasks performed by the crew (PF, PNF).

<u>UML modeling</u>: The output of this step is a description of the loop's tasks at three levels:

- loop level: the tasks achieved by the loop, typically assigned to specific steps
- step level: the tasks achieved by the steps

- task level: the decomposition of the tasks achieved by the steps, until the level of primitive tasks is reached.

A loop-based description of the tasks is therefore inherently hierarchical (loop ⇨ step ⇨ task). There is a *composition relation* between tasks. It is also usually very important to describe the *temporal relation* (order) between the tasks.

Tentative UML diagrams:
- activity diagrams
- state machine diagrams
- sequence diagrams
- use case diagrams

Remark: The HF-RTP also recommends (Deliverable D1.4) using ConcurTaskTrees (CTT) and the CASCaS architecture to model tasks.

2.3.4.2.5 Resources

These are the resources used by the AdCoS and allocated between its agents, see Figure 9.

To uncover the resources, proceed as for the tasks and consider the executive and adaptive loops in the AdCoS, and more generally the whole of the AdCoS loops control structure, see Section 2.3.3.3).
- Determine the resources used by the loops.
- Determine how the different loop steps may use specific resources (e.g., sensors), for the performance of the tasks assigned to the step.
- Determine how the possibly different sub-tasks within a step (especially if assigned to different types of agents contributing to the step) may require particular resources.

Many of the resources used by the loops are obvious and easy to observe. To perform resource analysis thoroughly the best approach consists in analyzing the tasks uncovered for the loops, steps, and sub-tasks within the steps, and determining which resource they require or use for their completion.

Applicable HF-RTP techniques & tools	
GreatSPN	University of Torino

For the commercial cockpit AdCoS example (Figure 17):

- Resources can be easily derived from the loops decomposition (into five steps) above. In particular, given the task of the executive loop is to control the A/C trajectory, the aircraft flight surfaces and a series of flight control computers are used as resources by the agents, as well as a series of sensors providing speed, altitude, position, A/C attitude, A/C angle of attack (AOA), etc.

UML modeling: The output of this step is a list of the resources used by the (agents in) the AdCoS (to perform their tasks).

To be very general, it's probably best to see resources as instances of processes (like objects, environments or agents), thus an object with a structure and some behaviors.

The UML diagrams for processes can therefore be used.

For resources,

Tentative UML diagrams:
- structure: component diagrams
- structure: model diagrams
- behavior: (behavioral) state machine diagrams
- behavior: sequence diagrams
- classes: class diagrams (to organize resources into class hierarchies)

2.3.4.2.6 Agents

To uncover the agents involved in the AdCoS, determine which agents are involved in the performance of the different loops' steps (executive and adaptive loops).

Determine the type of agent: human or machine.

Characterize each agent in terms appropriate for the analysis:
- *human agents*: Competences, skills, tasks they can perform.

- *machine agents*: Services, functions they can perform.

Dynamic agents can be modeled through various formalisms (e.g., state machines, Petri Nets, State charts etc. or cognitive models, in particular for human agents, though they can also be applied to modelling cognitive machine agents).

Applicable HF-RTP techniques & tools	
CASCaS	Cognitive architecture (OFFIS)
COSMO-CIVIC	Driver modelling (IFFSTAR)

For the commercial cockpit AdCoS example (Figure 17):

There are five agents
- PF (human)
- PNF (human)
- AP (machine)
- A/THR (machine)
- FMS (machine)

UML modeling: As stated above, agents in the proposed modeling approach are seen as instances of the more general class of processes (objects, environments, agents). The same UML diagrams can be used for all processes,

For agents,

Tentative UML diagrams:
- structure: component diagrams
- structure: model diagrams
- behavior: (behavioral) state machine diagrams
- behavior: sequence diagrams
- classes: class diagrams (to organize agents into class hierarchies)

2.3.4.2.7 Task distribution

Each (executive or adaptive) loop or step entail specific tasks that need to be performed (e.g., producing an action plan - action planning - to compensate for some discrepancies between a target and some actual value).

The modelling of task allocation consists of analyzing which agents perform which tasks and how. There are more or less two task allocation schemes:

- *static task allocation*: In that case, a given task is always performed by the same agent, or by an agent of the same family (e.g., capable of the same tasks, functions or services). This is the case in practice in many executive loops. In very advanced environments, the agents are very frequently specialized and only provide specific functions (that is for example the case in the control room use case, where specific human agents perform specific portions of the border monitoring control loop). In that case, task allocation design consists in deciding which agent will perform which task.
- *dynamic task allocation*: In that case, the same task can be performed by several agents. A given task may be performed by different agents over the course of AdCoS operations. Some mechanisms must therefore be defined to determine how the task is assigned to the agents. In the HoliDes AdCoS framework, dynamic task allocation between the agents involved in an executive loop is achieved through an adaptive loop: Task allocation within an executive loop is indeed one of the many parameter of that loop that may be controlled by an adaptive loop. In that case, task allocation design consists in designing the adaptive loop that performs the adaptation of task allocation (within the executive loop).

Machine assistance (i.e., assistance of human agent(s) by machine agent(s)) is a special case of task allocation: It occurs when a task can be split into subtasks and some of the subtasks are executed by one or more machine agents. The remaining subtasks are executed by one or more human agents. The human agent(s) are assisted by the machine agent(s) in the performance of the super-ordinate task.

Assistance is possible in both task allocation schemes:

- *static task allocation*: The subtasks are statically assigned to the machine and human agents, and the corresponding task sharing is static. It will not change over the course of AdCoS operations (e.g., navigation assistance systems).

- *dynamic task allocation*: The task sharing, i.e., specific allocation of the subtasks between the machine and human agents, can change dynamically over the course of AdCoS operations, possibly ranging from full manual control (human agents perform all subtasks) to full automated control (machine agents performs all subtasks).

The design of machine assistance systems - of prominent importance in many of the HoliDes use cases - is therefore a special case of the design of task allocation in cooperative human-machine systems.

Applicable HF-RTP techniques & tools
Seemingly none so far

For the commercial cockpit AdCoS example (Figure 17):
See loops decomposition in Section 2.3.4.2.3. It is easy to derive task allocation.

UML modeling: The outputs of this step are static or dynamic association tables between tasks and agents or vice versa (task ⇨ agent, agent ⇨ task).

Tentative UML diagrams:
- for static task allocation
  - association diagrams

- for dynamic tasks allocation: association diagrams in combination with:
  - state machine diagrams
  - timing diagrams (if temporal aspects are involved)

2.3.4.2.8 Resource allocation

Resource allocation describes how the resources available to perform the tasks (and complete the associated steps and loops) are allocated and used by the agents, included resource sharing.

Resource allocation resembles task allocation. It also is a correspondence problem: Resources are used by specific agents, or agents use specific resources. In terms of modelling, this consists in analyzing which agent needs which resource(s), or conversely which resource is used by which agent(s). This can be expressed in terms of correspondence tables.

Resource allocation may be static (when a given resource is always used by the same agent(s)) or dynamic (when the resource is used variably by multiple, changing agents, typically with mutual exclusion or at least resource access conflict problems. Some scheme must be defined to decide how the resources can be accessed, dynamically, by the agents). The mechanisms involved in the resolution of resource allocation conflicts (i.e., when two or more agents want to access the same resource(s)) can therefore also be described.

Analyze if resource allocation is adaptive. This concerns two aspects:

- *adaptation of the resources*: Adaptive resources are needed when the type and amount of resources used by the AdCoS are dynamically adapted, based on circumstances. For example, when the tasks dynamically assigned to an AdCoS are scaled up, meaning more difficult and resource consuming challenges must be addressed, it may be needed to adapt the resources dynamically.
- *adaptation of resource allocation*: Adaptation of resource allocation is when, beyond the mere adaptation of the resources themselves, the way they are allocated - or made available - to the agents itself is adapted. Given resource allocation can be seen as a correspondence problem (agent ⇨ resource(s), or resource ⇨ agent(s)), this simply means that this correspondence is dynamic and is adapted dynamically by a corresponding adaptive control loop.

Applicable HF-RTP techniques & tools
Seemingly none so far

For the commercial cockpit AdCoS example (Figure 17):
Resources are normally fully and completely available at all time (except in case of failure). Thus there is therefore no dynamic resource allocation.

UML modeling: Same as for tasks, the output of this step are static or dynamic association tables between resources and agents or vice versa (resource ⇨ agent, agent ⇨ resources).

Tentative UML diagrams:
- for static resource allocation
  - association diagrams

- for dynamic resources allocation: association diagrams in combination with:
  - state machine diagrams
  - timing diagrams (if temporal aspects are involved)

2.3.4.2.9 Interactions between agents (H2H, H2M & M2M)

Inter-agent interactions and communication modelling is about analyzing and modelling how the agents interact and exchange information, to satisfy the interaction and communication requirements associated with the operations performed by the agents (cf. determination of the AdCoS agents and task and resource distribution in the Sections 2.3.4.2.6, 2.3.4.2.7 and 2.3.4.2.8).

There are three categories of interactions/communications:

- human to human (H2H) communication: Design here is about deciding which information is communicated and how. Various interactions mean exist, natural or artificial, and imply various interaction modalities (visual, auditory).

- human-machine interaction (HMI) and user interfaces (UI) (H2M): Design is about defining how humans and machines interact and exchange information through human-machine interfaces, or user interfaces (UI).

- machine to machine (M2M) interactions: Designing is about defining how machine exchange information, through dedicated hardware and communication protocols.

To perform these analyses and produce the corresponding models, determine:

- How information flows *between the steps*, and in particular, if some tasks are to be assigned to different agents, how information flows between them (e.g., if agent A is in charge of perception and B is in charge of evaluation, a requirement must capture the need to circulate information from A to B).
- How information flows possibly *within the steps*, this happens when a step is shared by two different agents (human and/or machine) and information needs to be passed between them, for synchronization and cooperation.
- Information flows between agents, derived from the analyses above, allows modelling communication between the agents:
  - human-human communication (H2H)
  - human-machine communication (H2M) = human-machine interaction (HMI) and user interfaces (UI).
  - machine to machine (M2M) communication

  For each case, it is recommended to identify and model the communication modalities involved (e.g., vision, RF communications, etc.), the presence of interfaces (e.g. UIs, but also hardware interfaces for M2M, and any other characteristics relevant to interaction within the AdCoS).

As a second step, determine if communications and interactions are dynamically adapted:

- Interaction between the agents, determined above, may be adaptive: The means used to allow and support the interaction are dynamically adapted. This therefore concerns human to human (H2H) communication, human-machine interaction (HMI), including **adaptive user interfaces (UI)**, and machine to machine (M2M) interactions.

Applicable HF-RTP techniques & tools
Seemingly none so far

<u>For the commercial cockpit AdCoS example (Figure 17):</u>
See the "cockpit model" in Figure 17, it is easy to derive the interactions (H2H, H2M and M2M), including the user interfaces (shown on the figure), which resort from H2M.

UML modeling: The output of this step is a matrix of the interactions within the AdCoS agents, in terms of mode of communication (H2H, H2M, M2M), information flows, conditions in which they occur, etc.

Tentative UML diagrams:
- component diagrams (can be included in more general component diagrams for steps, loops and AdCoS)
- communication diagrams
- interaction overview diagrams
- information flow diagrams

2.3.4.2.10 Cooperation structure between agents

The cooperation structure describes the cooperation between the AdCoS agents. The cooperation structure can be analyzed:

- in terms of the set of cooperative units involved in the AdCoS. A cooperative unit is a set of agents performing dedicated sub-tasks that contribute to the same super-ordinate task. The composition of the sub-tasks is equivalent to the super-ordinate task. A cooperative unit can be modeled as a [T, [(A1, t1), (A2, t2),... (An, tn)]] where:
  - T is a super-ordinate task
  - A1, A2, ... are the agents involved in the super-ordinate task
  - t1, t2, ... are the sub-tasks assigned to the agents.
- in terms of a graph that characterizes how cooperative units intersect each other: A sub-task in a cooperative unit can be the super-ordinate task of another.

Determine if the cooperation structure is static or dynamic. The cooperation structure is dynamic when the assignation of agents to super-ordinate tasks in the cooperative units modeled above change (i.e., if the agents involved in a cooperative unit change) or when new cooperative units appear or when cooperative units disappear.

Applicable HF-RTP techniques & tools
Seemingly none so far

For the commercial cockpit AdCoS example (Figure 17):

Cooperation within the loops:

- executive loop

- AP & A/THR cooperate (if respectively activated by the PF/PNF).
- AP, A/THR and PF and/or PNF cooperate when flying in mixed mode.

- adaptive loop 1

- The PF and PNF cooperate in their management of the AP+A/THR.
- There is no cooperation between the PF/PNF and the FMS. They cannot be involved in the adaptive loop 1 at the same time. It is either the manual mode (PF/PNF) or the automatic mode (FMS).

- adaptive loop 2

- The PF and PNF cooperate in their management of the FMS.

UML modeling: The output of this step is a cooperation structure. It is analog to a communication or cooperation structure, with the added notion of tasks being shared.

UML diagrams similar to those used for interaction can be used.

Tentative UML diagrams:
For the interaction aspects:

- communication diagrams (may be redundant with interaction modeling above)
- interaction overview diagrams (may be redundant with interaction modeling above)

For the task sharing aspects

- collaboration use diagrams

2.4 HF Guidelines for Introduction of Adaptation

Adaptive Automation (AA) was introduced to overcome the disadvantages of static automation in complex Human-Machine-Systems (HMS). But also with AA the operator needs to be capable to perform the tasks assigned to him and to maintain a high level of situation awareness to achieve a high level of human-machine performance. To implement AA the tasks need to be well-designed and allocated to fit the human operator's cognitive capabilities as well as adequate feedback on the system's states needs to be provided [2].

The Human Factors (HF)-Guideline will provide a detailed orientation for the development process of the AdCos in HoliDes. The guideline considers human factors before, during and after the implementation of adaptive systems and AA into a cooperative multi-agent-system (humans and machines). Besides definitions from the literature the guideline will provide step-by-step introductions on how to consider human factors in an appropriate way.

In WP 1-5 different guidelines are created like the HF-ontology (WP1) and the communication-guideline (WP3). To achieve one holistic directive the HF-Guideline will include all created guidelines.

The appearance of the HF-Guideline will be in format of a handbook as first goal. After that the additional implementation into the HF-RTP, a website and/or a wiki is intended.

Main questions that will be addressed are:

- What is an **adaptive system** in terms of human factors-understanding?
- **Why** should HF be considered when implementing AA?
- What kind of **criteria** is decisive to allocate tasks?
- How should/can AA or dynamic function allocation be **implemented** (external/internal context)?
- What **HF aspects** need to be considered in respect to the models from WP2 when implementing adaptive features?
- How can an adaptive system be **tested**? What variables need to be considered? (Verification of the system's benefits)

To show the need of such a guideline a short definition of AA and a motivation why to consider HF when implementing AA into a human-machine-system is given.

2.4.1 Adaptive Automation in a HF Context

Deliverable D1.4 Chapter 2.4.1 already describes an approach called *Level of Automation* to optimize the assignment of control between the operator and machine by keeping both actors involved in system operations [3].

Another approach is labeled *Adaptive Automation* or *Dynamic Function Allocation*. It refers to the dynamic allocation of tasks between the operator and machine depending on situational demands [11]. In [12], the authors further distinguish between the two terms. Therefore *Dynamic Function Allocation* "may be defined as a scheme that may alter function allocation occasionally in time during system operation" (cf. [12]). The use of automation in this case is not automatically tied to situation demands but to operator demands. AA on the other hand is triggered by certain criteria to assure a safe performance or adjustment to the operator's workload [12]. AA functions are interactive aid working with the operator. On the contrary static automation can be seen as an agent working for the operator [2].

2.4.2. Criteria to activate automation in an adaptive system

The allocation of tasks between the operator and the machine can be determined by task demands, user capabilities and system requirements [2]. The criteria that need to be generated in order to allocate tasks in an appropriate way are assigned to four categories by [2]:

- Psychophysiology: evaluation of the operator's mental and physiological state
- Critical event logic: reaction to an environmental stimulus
- Model-based approaches: schedules the automation
- Operator performance measurement

In [1], the authors additionally refer to a hybrid logic that uses more than one category to activate automation.

2.4.3. Benefits and cost of adaptive automation

Implementing automation has benefits for the overall performance of the human-machine-system. It releases the operator from the tedium of routine tasks [2] and extends the functionality of the system (number of tasks and task duration). Apparent benefits from AA result from "sharing of control". The operator's capabilities are extended and supported when using automation. An example for the support is the power steering and power braking of a car. The operator can also be relieved from tasks so that his mental and physical capabilities can be spared. An example for that is a lane keeping assistant for driving. The operator can also be relieved from tasks when sharing subtasks with an automated system like steering the car and having the velocity managed by an on-board computer [12]. Due to the complex task environment static automation also has costs. The operator needs to monitor the system rather than to actively participate in the process [5]. The quantity and quality of mental workload change when working with an automated system [2]. Especially in multi-task environments where the operator has to perform manual tasks additionally to monitoring automation have negative impact on system performance. In [6], the authors have described this result as automation-induced complacency. Other resulting disadvantages can be the loss of situation awareness, impaired decision making and degradation of manual skills [2]. To overcome these disadvantages the approach of AA was introduced.

AA is intended to keep cognitive workload on an appropriate constant level. In times of low task load the operator takes over tasks, this prevents fatigue and maintains the operator's manual skills. When the workload it high, the automation sets in and enables a reduction of the cognitive workload [8]. By changing between e.g. manual control or low automation and high automation or full automation the operator is kept in the control. Therefore the operator deposes over cognitive capabilities to maintain situation awareness and therefore an improved decision making [7].

Another benefit resulting from changing between control responsibilities is the increase of the operator's vigilance.

AA can also have disadvantages that need to be considered early in the development process. A loss of situation awareness and performance degradation can occur when the operator is not aware of the system's current state. This effect can be evoked when a frequent alteration between phases of manual control or low automation and phases of high/full automation takes place. Poor communication strategies and feedback of a system can also result of a misunderstanding of automation's state. This can lead to "automation surprises" meaning automation behavior that was not foreseen by the operator. When the operator is surprised by automation, mistrust and distrust can appear leading to a higher workload. An additional problem resulting from state misunderstanding is a different or conflicting operator's intention of operation goals than the one the automation is targeting. This results in extra work that is not expedient for operation goals and leads to additional mental workload and impaired decision making.

2.4.4. System design requirements

One basic requirement to regulate the automation is the knowledge among agents about each other regarding current capabilities, performance and state [9].

Besides the questions guiding HoliDes "What should be automated?", "Why should it be automated?", and "Who should perform the adaptation?" (see *Deliverable D3.3*) it is important to determine criteria and rules to answer the questions "How should an AA communicate?" and "When should the automation occur?". The later question refers mainly to interruption management as it is described by [10].

Main points that need to be considered when implementing AA:

- Criteria to trigger automation
- Frequency of alteration between manual control/low automation and high/full automation
- Interruption management
- Task allocation
- Level of automation
- Communication strategies

The user interface, system design and implementation of adaptation are essential parts of the adaptive-cooperative human-machine system. The human *operator needs to be informed about the system status*. Therefore, the machine agents have to provide *intuitive feedback* and visualization to increase the operators understanding of the "*Why?*" and "*What?*" of the intended, anticipated or performed adaptation. For example, a machine agent within an Aeronautics AdCoS might communicate: "I display the fuel consumption for validation (the '*What?*') because we fly over a waypoint (the '*Why?*')".

Due to these reasons, some specific communication guidelines will be developed with the focus on the *communication strategies of the system adaptation*. The first version of these guidelines is in the Annex I Communication Guidelines.

2.4.5. How to approach adaptive automation assessment

With the HF Method Library developed in WP1 a comprehensive tool and method box to evaluate human factors is available to all project partners. The collection currently addresses the evaluation of usability, situation awareness, workload and user distraction.

3 Use Cases Specification

In this deliverable we have gone deeper in what are the scenarios of the chosen Use Cases exactly. We have focused on the kind of data we could collect and what is exactly the problem we want to solve.

3.1 UC1, Guided patient positioning

3.1.1 Use case characterization

The Guided Patient Positioning System consists of a the touch screen UI (User Interface) on Magnetic Resonance (MR) magnet that allows the operator to access various levels of information (e.g. more detailed instructions for novice users, more details of received physiology signals). The AdCoS user characterization is shown in Figure 18.

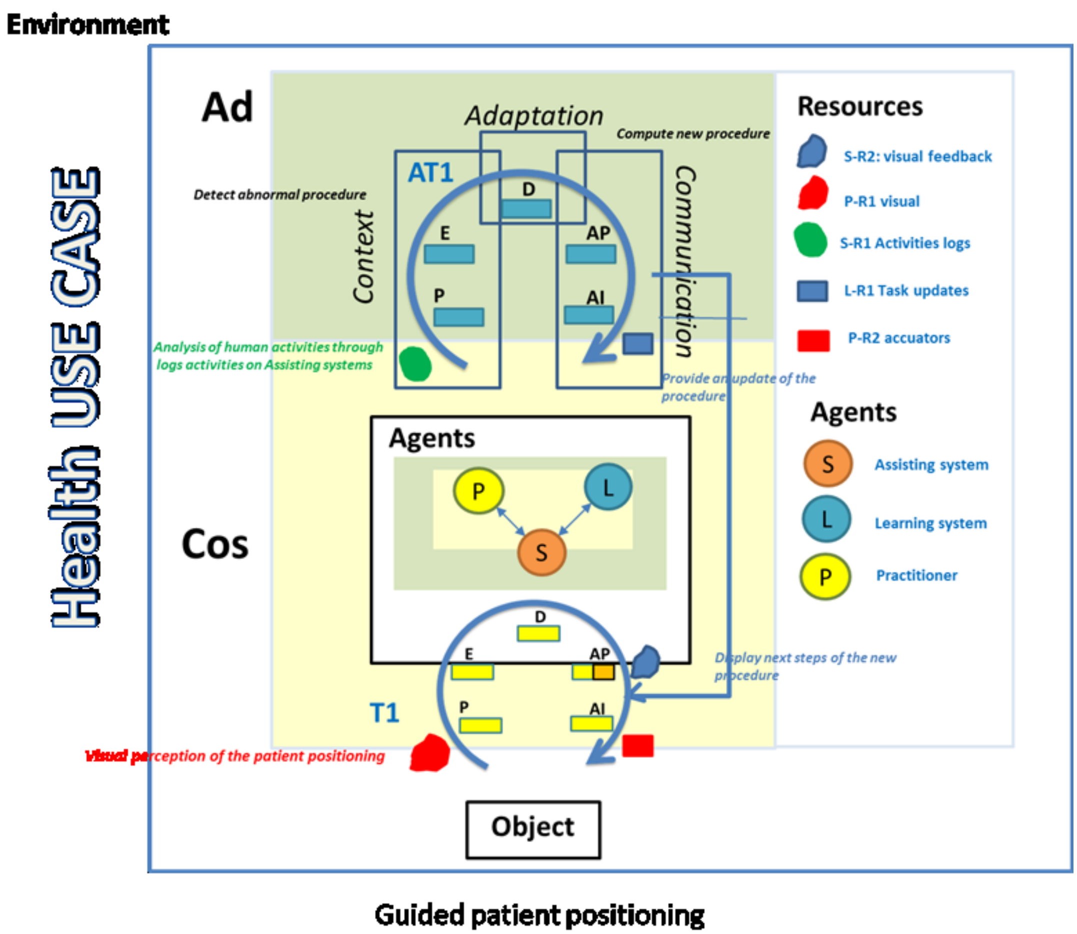

Figure 18: AdCoS characterization for Guided Patient Positioning (Health)

3.1.1.1 The environment of the AdCoS

The Guided Patient Positioning AdCoS interacts with its controlled entity, the operator and the external environment as outlined in the following.

Controlled entity

The controlled entity in the case of the Guided patient positioning AdCoS is Magnetic Resonance Imaging (MRI) scanning scene – the scanner and the patient. In this scenario, the scanner and the patient are equally important in the creation of the necessary imaging.

The aim is to create a situation (a state) which is functional to capture the right image and safe and comfortable for the patient.

In this architecture, the operator works to manage the scene and the AdCoS follows this work, giving advice along the way.

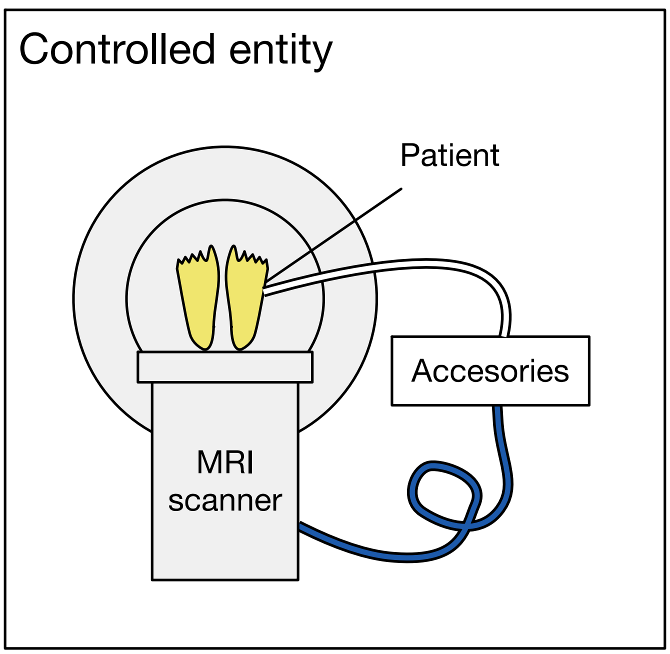

Figure 19: The elements making up the controlled entity of the Guided patient positioning AdCoS

Operator of the AdCoS
The user is the MRI operator; s/he wants to be sure that the examination he selects and performs for a patient is the examination that the radiologist ordered. The operator has to be confident that the image quality for the radiologist is up to standards, within the time available and with high patient comfort. The operator often works with another operator where one positions the patient while the other is in the control room. The operator that will be positioning the patient will be the user of the AdCoS. Here the operator ensures that the patient and coil are positioned correctly and the patient is comfortable.

External environment
The most important element of the external environment is the control system of the scanner, which is coincidentally an AdCoS itself.

The external environment provides the information that makes up the external context of the scanning and the starting point of the guidance for the positioning procedure.

3.1.2 Scenario Detailed Specification

3.1.2.1 Details on AdCoS and environment

The AdCoS shall provide on-line guidance and actual information during positioning of the patient. It needs to use the input data:

- Current patient (name, age, weight, etc.)
- Special patient characteristics (pregnancy, implants, etc.)
- Clinical request
- MRI examination procedure
- Connected coils
- Connected accessories
- Signals received from accessories, if applicable (e.g. ECG signal)
- Environment conditions (temperature, humidity, etc.)
- System settings (e.g. setting of headset volume, ventilator, light)

From the patient characteristics and MRI examination procedure the system can derive the instructions for the operator, which need to be updated on-line based on detectable actions by the operator. Additionally sound can be used to provide feedback to the operator.

The system shall provide clear and timely feedback to the operator on the status of relevant connected accessories, and help the operator in making corrections if necessary.

The system shall support multiple users, since patient positioning might be performed by more than one operator. Also other medical staff might be present, e.g. the anaesthesiologist.

The system could use historical data, e.g. derived from the systems log-file, to predict the flow of actions and optimize guidance.

Operator's background:

MR operators have been more extensively trained and educated in the broader aspects of MR safety issues, including issues related to the potential for thermal loading or burns and direct neuromuscular excitation from rapidly changing gradients. Optionally MR operators have been further trained in MRI through an accredited program. On top of this, MR operators typically receive device specific training, which is also the case for operators using Philips equipment. Philips offers a device specific training by a Philips Clinical Education Specialist.

Nevertheless, operators may switch between modalities (MRI, CT, X-ray) and between MRI systems of different vendors. The number of available operators is reducing at several regions, so the work needs to be done by less staff and the education and experience level will vary.

Operator actions during patient positioning:

Connection of coils and accessories:

- RF coils: For optimal image quality the right RF receive coils need to be connected and positioned.
- ECG electrodes: In case the scan needs to be synchronized with patient's heart rate ECG electrodes need to be attached to the chest and the ECG detector needs to be connected.
- Respiratory sensor: In case the scans need to be synchronized on the breathing of the patient a respiratory sensor needs to be attached to the patient chest.
- Audio: Apply ear plugs and headset to prevent hearing damage, to allow communication with the patient and to provide music to the patient during the examination; Adjust audio volume

- Provide nurse call balloon to the patient to allow to signal or alarm the operator, and instruct the patient how to use this.
- Comfort and immobilization: In order to prevent movements of the patient during scanning pillows and straps may be used to prevent motion and to arrange a comfortable position.
  Adjust patient ventilation and light settings, if necessary.
- Patient instructions: Instruct the patient on "do's and dont's" during the examination.
- Scan plane: Determine the centre of the anatomy to be scanned and shift the patient into the system to the preferred location.

<u>Safety risks to be reduced by the Guided Patient Positioning System:</u>
During positioning of the patient, it is very important to be aware of all safety related aspects:

- RF heating:
  - Avoid loops of body parts (e.g. calves of the legs too close together, hand-in-hand position)
  - Avoid arms positioned too close to the side of the bore
  - No loops in conductive wires of coils and accessories (like VCG leads)
- Ventilation:
  - Provide adequate ventilation and do not cover the patient too much
- Peripheral nerve stimulation:
  - Explain the patient that he might encounter peripheral nerve stimulation due to fast switching of magnetic field gradients
- Acoustic noise:
  - Provide adequate hearing protection (for adults in-ear plugs and headset)
  - Adjust intercom volume on headset for voice instructions to the patient
- Table movement:
  - Prevent pinching of patient parts (e.g. fingers)
  - Prevent clamping of clothing / blankets
  - Avoid pinching / clamping of leads (e.g. of ECG), wires (e.g. of coils), or tubes (e.g. of nurse call and respiratory sensor)
- Nurse call:
  - Provide the nurse call to the patient and explain that this needs to be pressed once to get attention and twice to generate an alarm

3.1.2.2 Scenarios

The mean-ends model can be used to derive detailed scenarios. This model organises the goals, functions and behaviour on five levels, in a flat hierarchy.

The top and second level (goal and objectives in the model produced here) consist of the elements listed in Table 5 below.

Goal
Provide safe, functional and comfortable situation for MRI scanning
Objectives
Provide patient in correct physical situation for scanning
Provide safe transmission channels for radio frequency signals
Provide physical environment to acquire image
Provide passive safety for patient
Provide means of communication
Provide a comfortable environment

Table 5: Elements of the model at the goal and objectives levels

Due to the flat hierarchy, the diagram will have to be split up for the purpose of the presentation in this document. In the following figures, the objectives listed above are further divided into functions and the associated behaviours are identified.

In Figure 20 to Figure 23, the lowest level (on a grey background) contains the mapping to the actions identified as a part of hazard analysis outside of the scope of HoliDes.

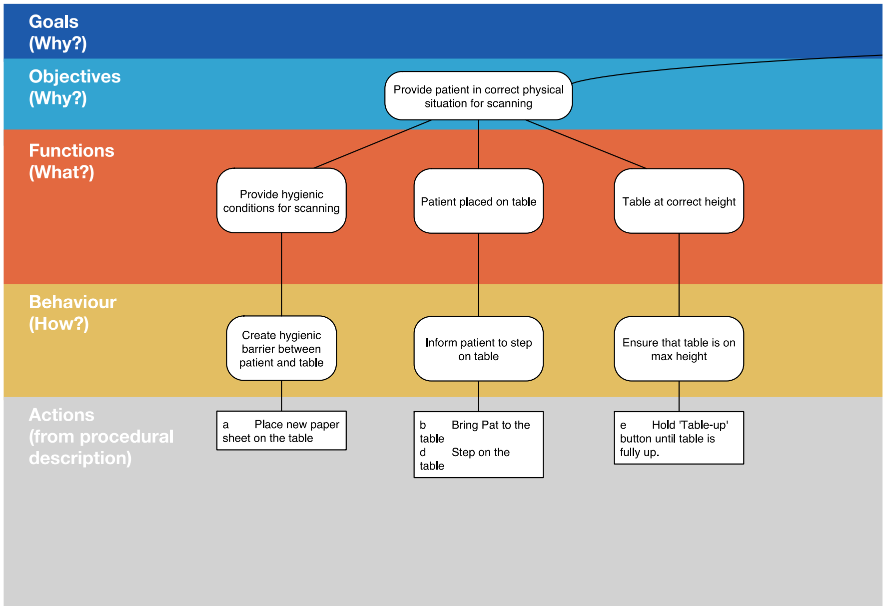

Figure 20: Expansion of the objective "Provide patient (pat) in correct physical situation for scanning"

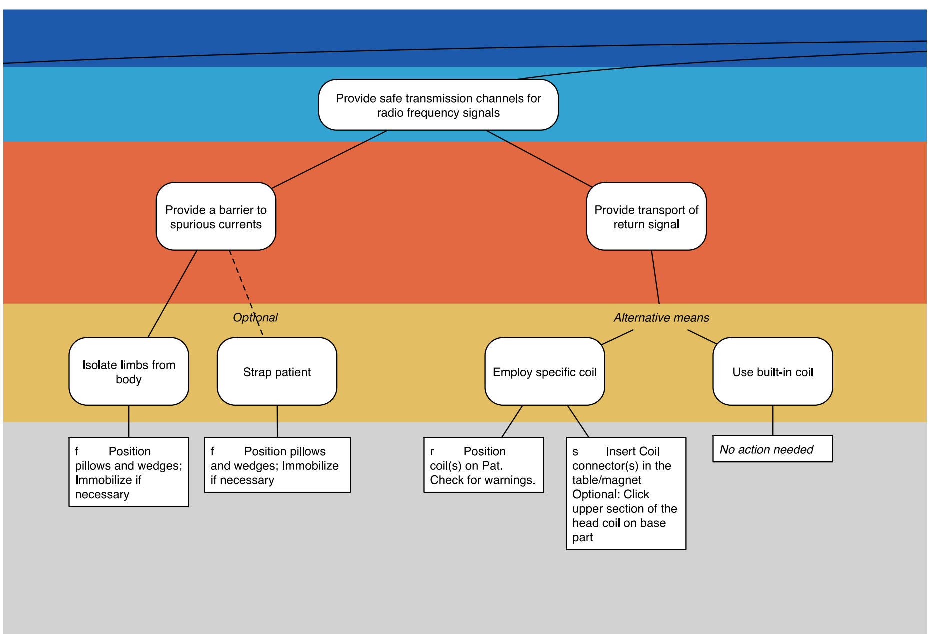

Figure 21: Expansion of the objective "Provide safe transmission channels for radio frequency signals"

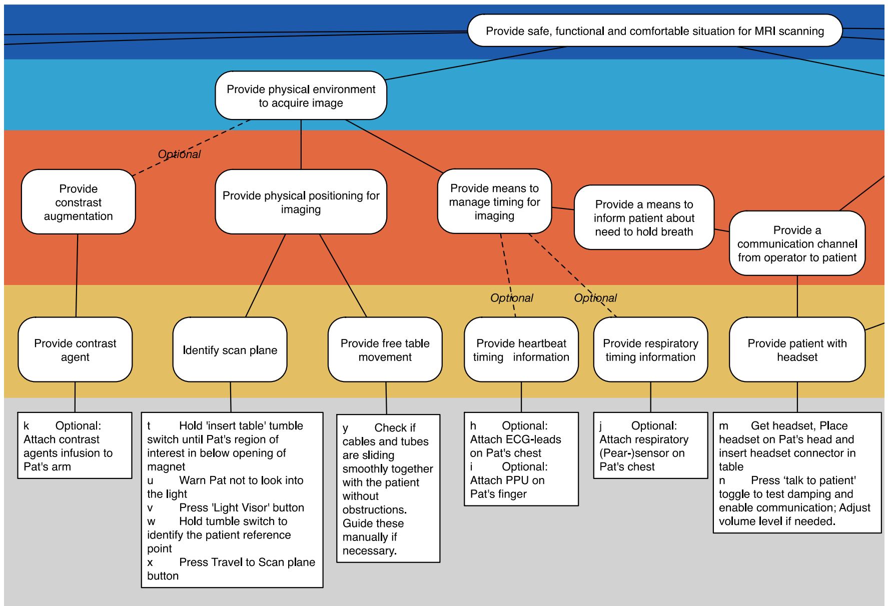

Figure 22: Expansion of the objective "Provide physical environment to acquire image"

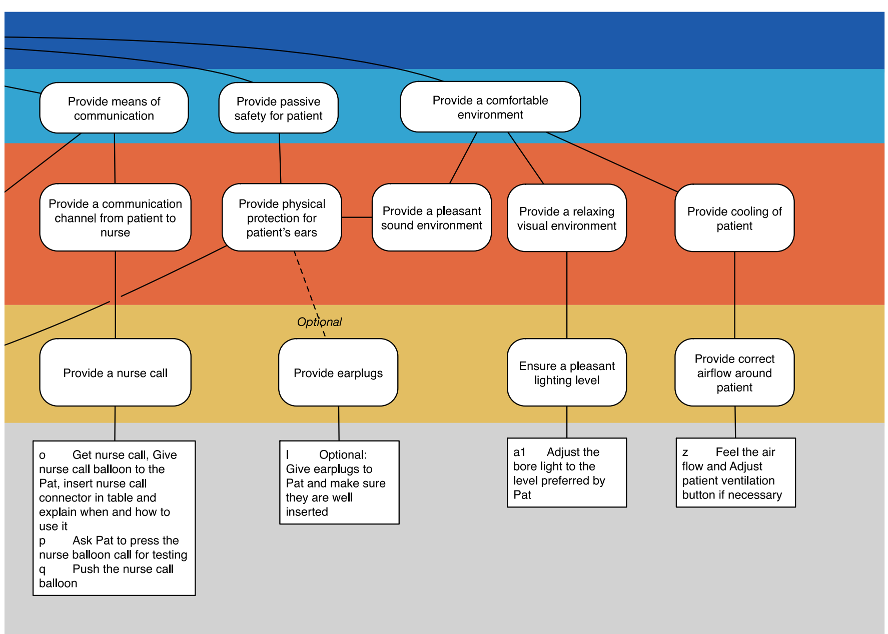

Figure 23: Expansion of the remaining objectives

3.2 UC2, Diversion Airport

3.2.1 Use case characterization

The system's division in 'Ad' and 'CoS' part. In normal operations the systems is responsible for keeping flight relevant information up-to-date. If pilot or situation induces deviation from the original destination, the 'CoS' acts as information integrator to create a relevant situation model and via performing adequate calculations the 'CoS' communicates diversion options to the pilot. As the evaluation of diversion options is a complex task, the 'Ad' part of the system assesses pilot's mental state and if the state is determined to deteriorate, 'Ad' triggers the adaptation in 'CoS'. Details on the two parts are in the following paragraphs.

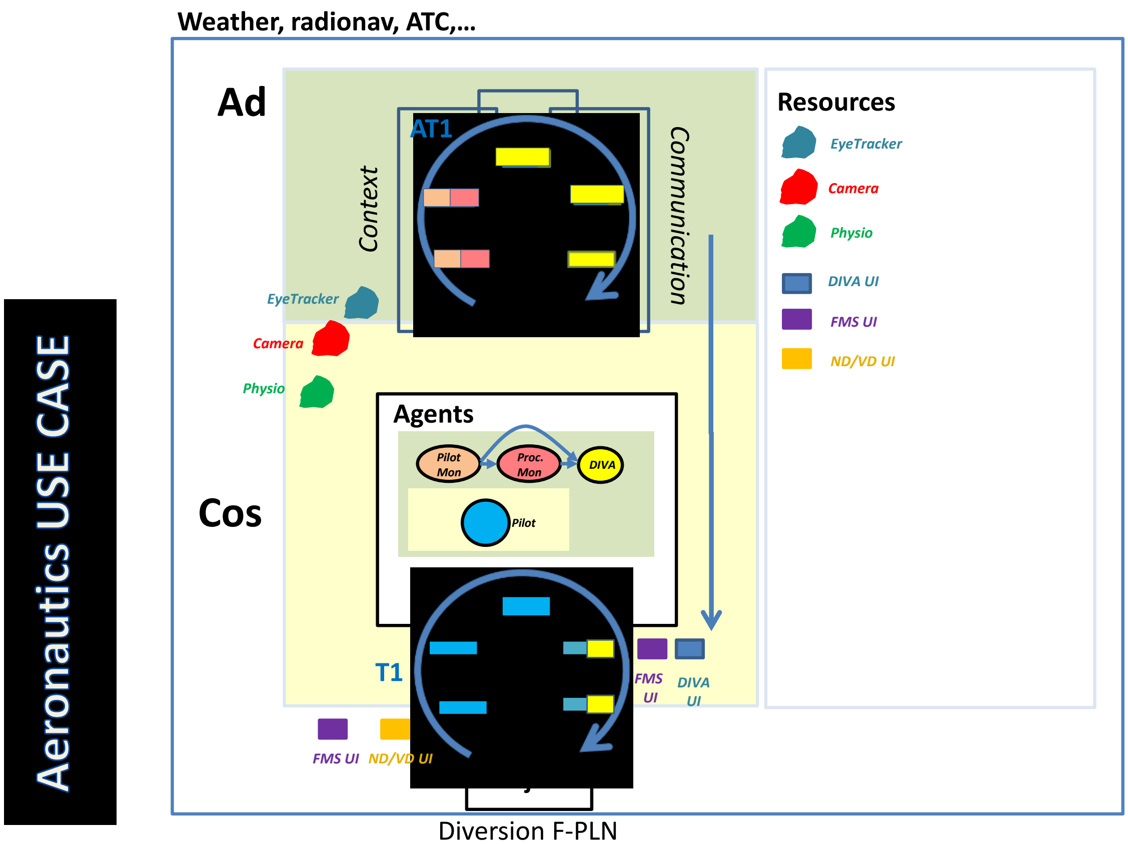

Figure 24: AdCoS characterization for UC2 (aeronautics). The system consumes and processes data from several sources of information.

Adaptation is triggered based on data measured on the pilot in order to provide optimum presentation of the processed data.

3.2.1.1 The 'CoS' part

Since the definition of parts of the CoS – the pilot, the pilot monitoring system, the procedure monitoring system and the diversion assistant – in the Deliverable D3.4, the role of each part and their interactions have been elaborated.

The pilot is responsible for flying, navigating and communicating. When a diversion is required, the following steps needs to be performed:

1. Securing aircraft – The pilot brings the aircraft into holding. While in cruise, it is a simple maneuver, but close to the original destination, the pilot needs to find and plan route to a defined holding path.
2. Updating and integrating information related to the original destination, alternate or diversion destinations. The information covers as diverse topics as aircraft performance or weather situation.
3. Based on the situation model created in the previous step the pilot decides about the diversion destination and communicates the decision to ATM. If cleared for the new destination he re-plans the flight to the new destination.

The diversions usually take place at the end of flight; that is in situations when pilots are subject to fatigue, stress and when they rely most on the memory with respect to the overall flight model they created before starting the flight and later just updated with the available information. The pilot and procedure monitoring reveal potential degradation of performance due to fatigue or misalignment between pilots' situation model and reality. Based on the analysis of available methods to assess the state of deterioration, several physiological markers of stress, workload or fatigue will be integrated in a machine learning classifier. The markers will read EEG, ECG and eye-tracker data.

DivA (Diversion Assistant) is supposed to provide assistance during the diversion. The most important aspect of the assistant for diversion is to obtain the latest information on the overall situation and to complex calculations including flight path optimization and selection of the optimal strategy based on heterogeneous factors. With respect to the adaptation, DivA is the effector of adaptation.

3.2.1.2 The 'Ad' part

The adaptation in the DivA UC has its triggers, the information about pilot's psychophysiological state and his performance, and its effector, the DivA assistant system.

The strong accent on safety and determinism in aviation requires that adaptation needs to be well justified and it must be:

- Comprehensive to pilots – They need to understand why adaptation was invoked and what has been adapted.
- Consistent – Adaptation is realized in the same way in the same or similar situation.
- Unobtrusive – As little as possible should be changed.

The EFB (Electronic Flight Bag) platform is less restricted by regulations, therefore it may be a gate for adaptation in the cockpit. Diversion Assistant (DivA) AdCoS will have to adapt the prioritization in selecting and evaluating available airports.

DivA will implement two modes of adaptation. First, in reaction to the changes in environment (illumination, turbulences etc.) it will adjust display properties to ensure readability in all conditions.

Second, in reaction to the state of the pilot, DivA will adjust prioritization in selecting and evaluating available airports. The factors that are considered by DivA can be grouped in five categories:

- Fuel related factors (the economy of flight and the range)
- Availability – Weather factors and NOTAM (Notice to AirMan messages modifications to aviation infrastructure
- Airline preferences for various airports based on existing contracts and available services (the economy factors)
- Crew limiting factors – Licensing
- Aircraft limiting factors – What airports can be used for given aircraft type

If the state of the pilot deteriorates, a more conservative strategy focused on safety rather than economy will be applied. This adaptation is intentionally hidden from the pilots.

3.2.1 Scenario Detailed Specification

The adaptation scenario consists of three steps – context assessment, system adaptation and communication of adaptation to the human operator.

In the DivA AdCoS the context assessment assumes the determination of the pilot's mental state with respect to attention loss and/or fatigue. Two tools from HoliDes HF-RTP are used – missed event detector (MED) and pilot state classifier. MED interprets camera recording with respect to expected procedures. Pilot state classifier evaluates physiological data for patterns related to onset of fatigue.

The adaptation evaluates data from MED and pilot state classifier. When a deviation from expected performance is detected, the DivA AdCoS applies one of three adaptation strategies: adjustment of weights in safety-economy trade-offs during prioritization of available airports, changes in persistence and saliency of displayed information and adjustment of display properties to match environmental situation.

The communication to pilots depends on the selected strategy – inherent changes in the algorithms hidden from the pilots, or changes in displays. However, the aim is to minimize the extent of changes in the cockpit.

3.3 UC3, Command and Control Room

3.3.1 Use case characterization

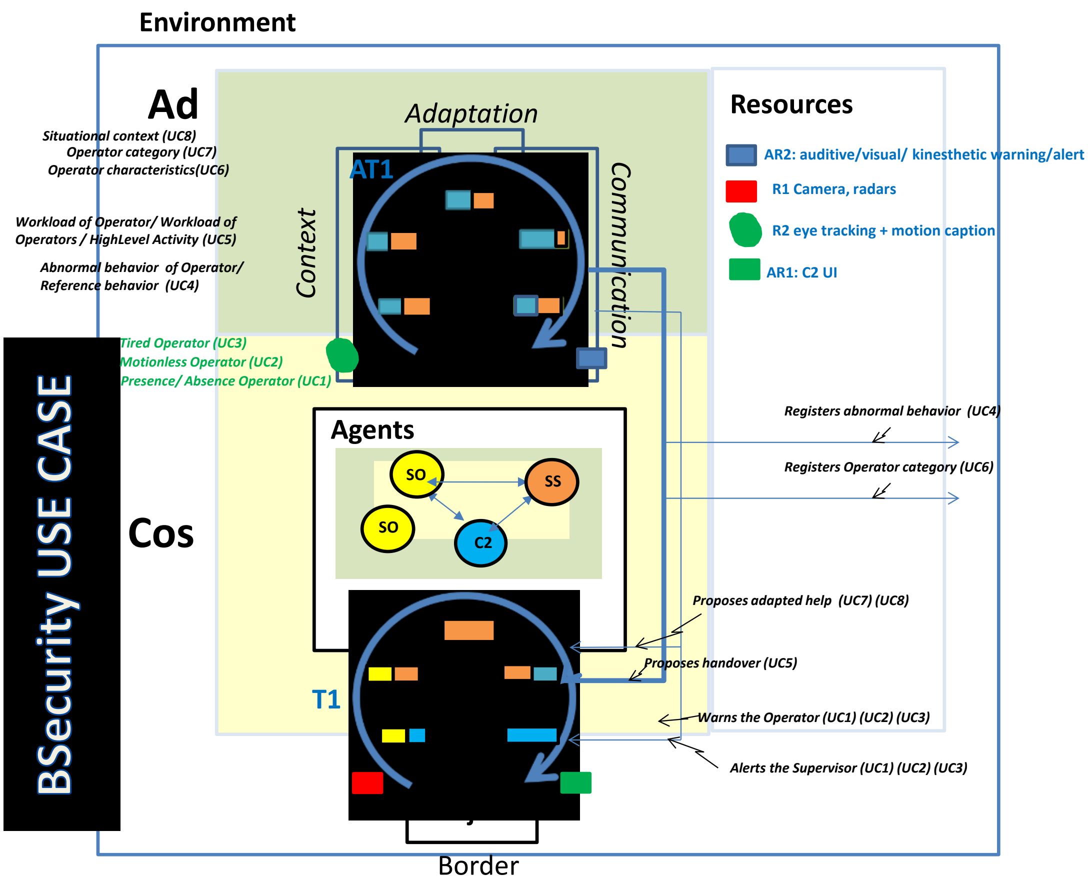

Figure 25: AdCoS characterization for UC3 (Border Control Room)

The Command & Control rooms includes

- several Surveillance operators
- one Surveillance Supervisor
- The Command & Control (C2) Information system, including sensors (Radars and Cameras) and Common Operational Picture (COP) display.

All of them, are facing personal screens, and work together on a single object: the border and its security. They do so in the environment in which the Command & Control (C2) Information system operates, and have to detect human and non-human intruders. The situation encompasses normal (no events, or no threat events) and critical situations (with threat event). They have to provide inputs into the C2I system. Common Operational Picture (COP) supports the human operators' situation awareness (SA) in the Control Room. Additional support is available to the operators who execute control tasks in form of alerts, recommendations (e.g. presentation of predefined tasking orders to resolve critical situations) and facilities for evaluating potential threats.

The work of operators in a border security control room is characterized by alternative phases of increased activities during events to be handled in a reduced time and of low-workload with little or no activities during long periods of time.

The border control room AdCoS attempts at increasing the effectiveness of the border security organisation by ensuring that operators are available at their workstations and are ready to operate efficiently when needed. It includes several Uses Cases (UCs).

From the potential UCs initially mentioned in the Figure 25 (UC1 to UC8) initially proposed for adaptation in Deliverable D3.4, and coming from WP8 owners, better criteria emerged for assessing the quality of UCs in terms of adaptation, and we decided, in accordance with the AdCoS owner, to focus especially on WP8's UC1 to UC5. To be coherent with the numbering of the Use Cases within WP3, we use below a sub classification within Adaptation UC3 to define the different Use Cases concerning WP8 Command & Control rooms. They are summarized below:

Sub Use Case 3.1 Operator Absent from Work Place

An operator is absent from his workplace for a longer than accepted period of time. The system calls the operator back to his workplace. If he does not return to this workplace after a defined length of time, his supervisor is informed.

Sub Use Case 3.2 Operator Idle at Work Place

An operator is present at his workplace but idle for a longer than accepted period of time (idle is defined as motionless suggesting that the operator is asleep). The system contacts the operator. If he does not display any activity after a defined length of time, his supervisor is informed.

Sub Use Case 3.3 Operator Tired in Work Place

An operator is present at his workplace but displays signs of fatigue. The system contacts the operator with a warning. If he doesn't acknowledge the warning after a defined length of time, his supervisor is informed.

Sub Use Case 3.4 Registration of Unusual Operator Behaviour Patterns

Individual and cumulative instances of operator absence can be plotted with the aim of allowing the border security management to identify behaviour patterns of the crew that can be exploited by third parties in order to compromise a station's security.

Sub Use Case 3.5 Load Balancing on Operator Level

The system is able to recognize the load of a single operator compared to the overall load of all operators in one headquarters. To avoid overloading an individual operator, the system shall distribute incoming events to operators with a lower current workload and offer the redistribution of events from operators who are dealing with a number of events above a critical threshold.

3.3.1 Scenario Detailed Specification

The system is able to recognize the state of individual operators and initiates a response if a measured state is outside of the allowed range. The physical and mental states covered by the system are:

- Presence/absence of the operator from his workplace at a given point in time or for a given period of time;
- Lack of movement for a given time of an operator present at his workplace, suggesting that he is asleep;
- Particular behaviours that suggest tiredness and/or lack of concentration.

All the results of the above methods must be surveyed by the system constantly. If any change of state takes place and is noticed by the system, the system takes one of the following actions:

- Motivate the operator to remedy the situation (return to his desk or take measures to overcome his fatigue).
- Notify the supervisor about an operator status that could interfere with proper system operation.

Suggest measures (replace sleepy operator, transfer tasks away from him to resolve overload) or initiate a workflow (e.g. process transfer) to resolve the situation (see Figure 26).

The decision on how to handle such a situation must be taken by the superior officer. The system can only provide hints or workflows to assist him to find the appropriate solution.

In addition the system logs all instances of absence / tiredness in an anonymized way in a database that is analysed in regular intervals for detecting unusual patterns that could be possibly exploited by perpetrators monitoring those behaviours.

Some details in the Sub UC3.5 Load Balancing on Operator Level are presented in the followings:
The system must be able to detect the workload of each individual operator. This could be done taking into account a number of variables:

- *Number and criticality of current events:* the number and characteristics of the events the operator has to handle.
- *Level of experience of the operator:* Operators may by assigned one of three levels of expertise (e.g. 'Basic Experience', 'Advanced Experience', 'Expert Experience') with experienced operators being expected to be able to simultaneously handle a number of critical events.
- *Level of fatigue of the operator:* The operator's tiredness as measured by sensors can reduce his expected performance level.
- *Time the operator needs to perform his tasks:* This could be measured by comparing a "standard" workflow time schedule (including certain "milestones" within the schedule) with the actual time needed by the operator for running through the workflow. The longer he takes to

perform the actions (or the more often he misses the milestones) the closer he may be to an overload scenario.
- *Number of errors produced by the operator:* Errors may be due to stress caused by an overload scenario (e.g. presses of back/undo buttons, erroneous input leading to error/warning messages).

The system identifies the following limits/thresholds for workload levels:
- Idle / low level of workload
- OK (optimum level of workload)
- Pre-overload
- Overload

If the overload limit is reached, the supervisor is notified by the system informing him that one of the operators is working inefficiently because of a high subjective workload. The system proposes to re-assign one or more events to another, less busy operator. The supervisor then decides which actions to take. Most likely, he will transfer one or more events to other operators. Therefore, he must be informed about the individual workload of each operator. The system implements a task transfer process that takes into account the individual workloads of all other operators that might possibly take over the task(s). The process should suggest the operator to take over the task (e.g. the one with the lowest workload) and the task(s) to transfer (preferably tasks that are in a defined state and do not require additional interaction between the operators in order to be transferred including all information related to the task). This process is illustrated in Figure 26.

In case all operators are under overload, a load balancing procedure on headquarter level must be applied (this adaptation is not developed as part of HoliDes).

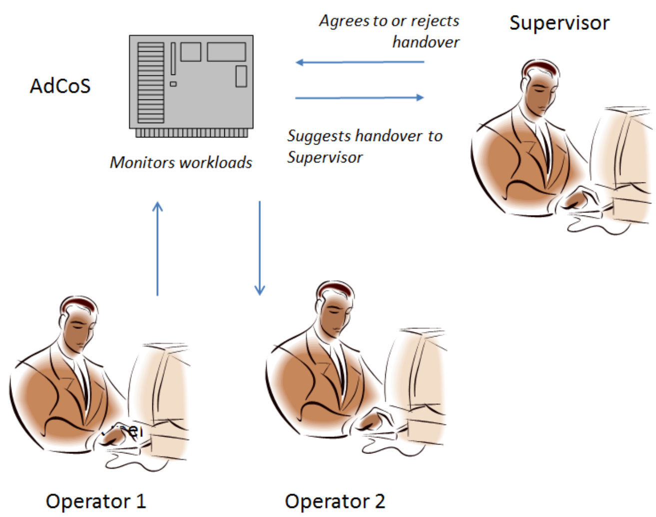

Figure 26: Schematic overview of the workload balancing use case

To reach and provide a solution for this AdCoS, the three main functions that are developed within WP3, context assessment, adaptation and communication take part of the solution, by participating in the different Sub-Uses Cases, as followed.

In terms of **context assessment**, WP3 participates in to the following Sub-UC:

- **Sub UC 3.1 Operator Absent from Work Place**: *Presence:* Sensors register absences of operators that are longer than a permitted time,
- **Sub UC3.2 Operator Idle at Work Place**: *Movement:* Sensors register the lack of movement of operators present at their workstations,

- **Sub UC3.3 Operator Tired at Work Place** *Fatigue:* Sensors register and calculate the symptoms of sleepiness in operators.

- **Sub UC3.4 Registration of Unusual Operator Behaviour Patterns**: Data captured on presence, movement and fatigue are collected and analysed to determine patterns that could be observed and exploited by perpetrators.

- **Sub UC3.5 Load Balancing on Operator Level***:* The system calculate the current subjective workload for every operator (including level of experience, training levels achieved, number and criticality of current events to be handled, and others).

In term of **adaptation,** WP3 contributes to the following Sub UC:

- **Sub UC3.4 Registration of Unusual Operator Behaviour Patterns***:* From detect patterns; the adaptation will take place at organizational levels to reduce risk of attacks.

- **Sub UC3.5 Load Balancing on Operator Level***:* the system calculates the optimal balancing between operators and proposes to the supervisor re-assign individual events from high subjective-work loaded operators to low-workloaded operators.

In term of **communication,** WP3 participate in the following Sub UC:

- **Sub UC3.1 Operator Absent from Work Place***:* If operators are absent for longer than the permitted time, the system calls them back to their workstations by means of discrete actuators worn by the operators.

- **Sub-UC3.2 Operator Idle at Work Place:** If operators display a lack of movement for a longer period of time, the system wakes them by the same means as Sub UC3.1.

- **Sub-UC3.3 Operator Tired at Work Place***:* If the operator displays symptoms of fatigue, the system suggests to the operator to take appropriate measures.

- **Sub UC3.5 Load Balancing on Operator Level***:* The re-assigning individual events will be supported by appropriate information of the operator and of his supervisor.

3.4 UC4, Overtaking including lane change assistant

3.4.1 Use case characterization

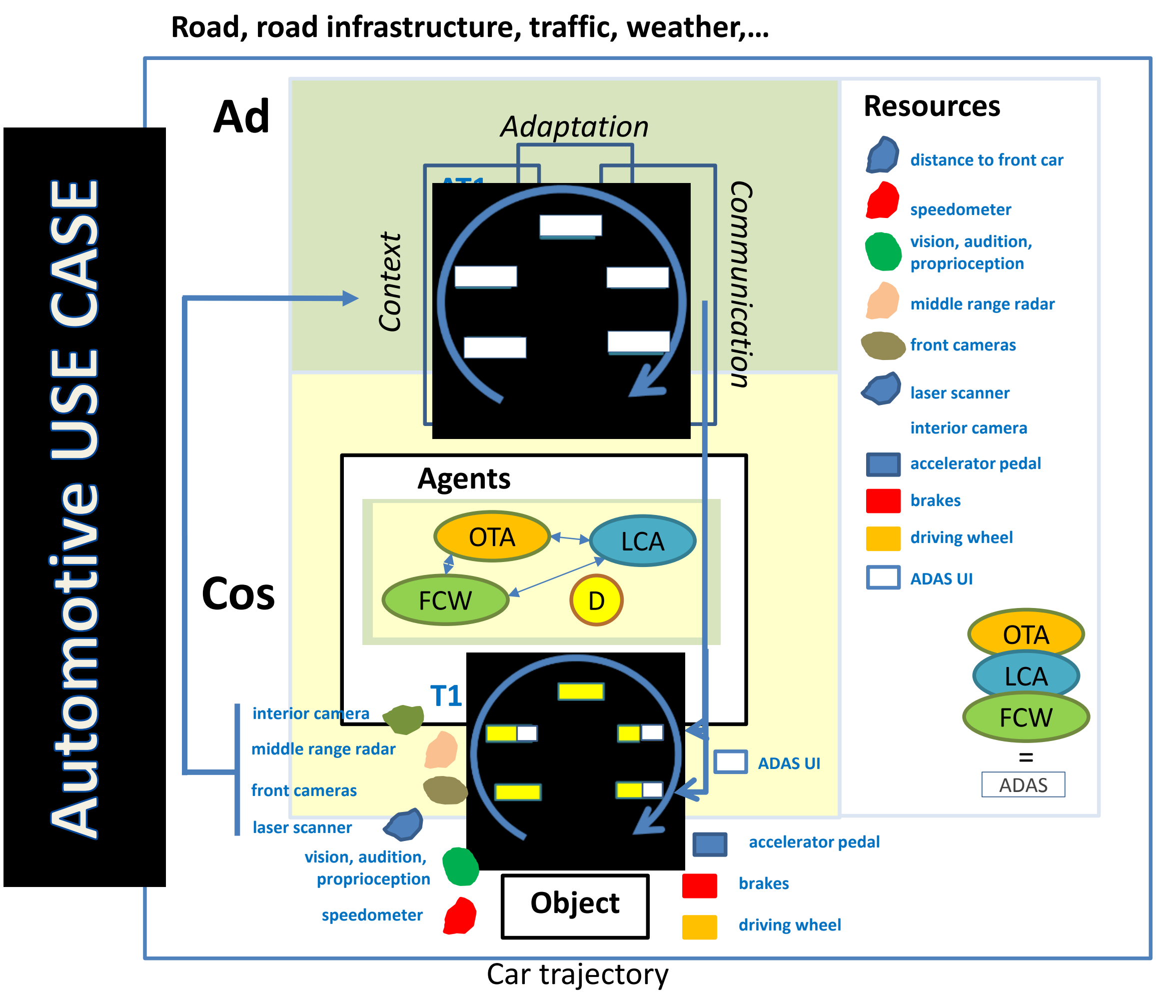

Figure 27: AdCoS characterization for UC4 (automotive)

UC4 "Overtaking including lane change assistant" was already described in *Deliverable D3.4*. The focus of the use case is an overtaking maneuver to overtake vehicle C while vehicle B is approaching from behind. As already mentioned in *deliverables D3.4* and *D9.3* for vehicle A different levels of automations namely "Assisted/Partial Automation" and "Conditional Automation" are addressed and represented by different prototypes in WP9. Therefore we will divide the use case specification later on into two parts "Adapted Assistance" and "Adapted Automation".

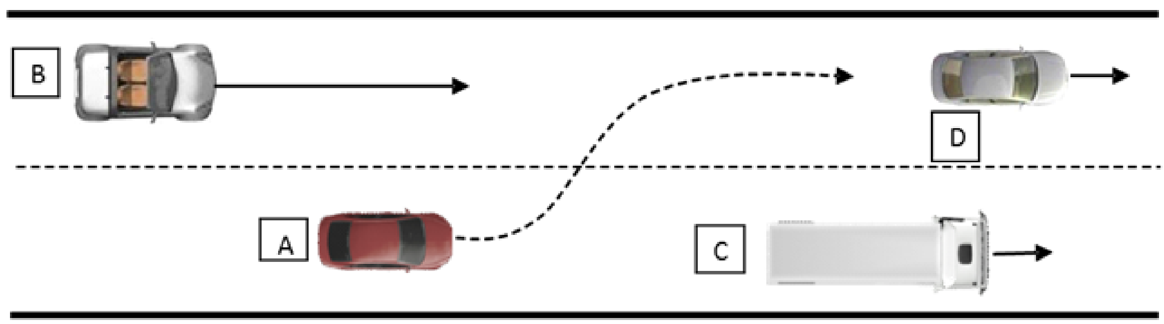

Figure 28: Overtaking Scenario on a Highway

Nevertheless both parts can be described by a common AdCoS characterization as shown in Figure 27.

Figure 27 recalls the characterization for UC4 derived in *Deliverable D3.4*. Compared to *Deliverable D3.4* we only made a minor change and added a forth machine agent: the driver model "DM". We added the driver model to highlight explicitly the part of the co-pilot which is responsible for analysing the internal context of the driver (driver state, manoeuvre intentions, driving style). The information about the internal context is used as input for the other three agents of the co-pilot and therefore directly influences their behaviour. In other words the introduction of the agent "DM" helps to explain and visualize the process of the adaptation in UC4 as we will see in the next section.

Since the driver model is still a part of the co-pilot and the assistance system no changes in the description of the Ad and the Cos part compared to *Deliverable D3.4* are necessary. Therefore we refer to *Deliverable D3.4* for a detailed description of the use case characterization.

3.4.1 Scenario Detailed Specification

In this section we specify in more detail the information flow between the different agents within the use case. We will explain the information flow by using the concept of sequence diagrams. Moreover we will point out what part of information belongs either to the context assessment block or the adaptation block or the communication block. Additional we will link the used WP3 tools and techniques to the blocks.

As already mentioned above the automotive use case considers different levels of automation. Therefore we will divide the use case specification into two parts "Adapted Assistance" and "Adapted Automation".

Adapted Assistance

For the specification of the use case we consider different scenarios of the use case. As mentioned in *Deliverable D9.3* these scenarios assume a variation in the driver state (driver state okay/not okay, for instance distracted through cell phone or navigation system) and a variation in the environmental state (lane change possible/not possible). Figure 29 to Figure 31 illustrate the sequence diagrams for these scenarios.

Green boxes and arrows show the external context parts in the scenario. The external context relates to information about

- recognized objects (surrounding traffic participants and traffic signs)
- the future path of the road including
  - the current vehicle state like e.g. current velocity and acceleration
  - the current state of the actuators like e.g. steering wheel, angles and pedal positions.

The blue boxes and arrows address the internal context. These boxes indicate that for the internal context the WP3 tools Movida, Bad MoB and Distraction Classifier are used (see *Deliverable D3.3* and *D.9.3* for more details). More precisely Movida and Distraction Classifier cover the context information driver state (visual distraction). Movida and Bad MoB cover additionally the context information about the performed behaviour and the intended behaviour. These tools are used to implement and realize the adaptation of the use case.

The adaptation of the use case is not directly visible as a box in the sequence diagrams, rather it is indirectly visible through the existence of different possible scenarios. For that reason the adaptation covers an adaptation of the HMI. Based on the driver state and drivers behaviour/intentions different warnings and information in the cockpit display are presented. Moreover the adaptation covers an adaptation of the control task. In critical situations the machine agents LCA, OTA, or FCW will take over the control to guarantee a safe driving.

The communication in the use case is visualized through orange boxes and arrows in the sequence diagrams. The communication of the adaptation directly depends on the output of the context assessment as we see in the sequence diagrams. The different aspects of the communication pointed out in *Deliverable D3.4* can be clustered in direct and indirect communication. Direct communication is provided by visual information and warnings in the cockpit display (see Figure 32). Indirect communication is provided through the action of the machine agent on the vehicle control and the movement of the vehicle and steering wheel as a consequence. Both types are addressed by the different scenarios.

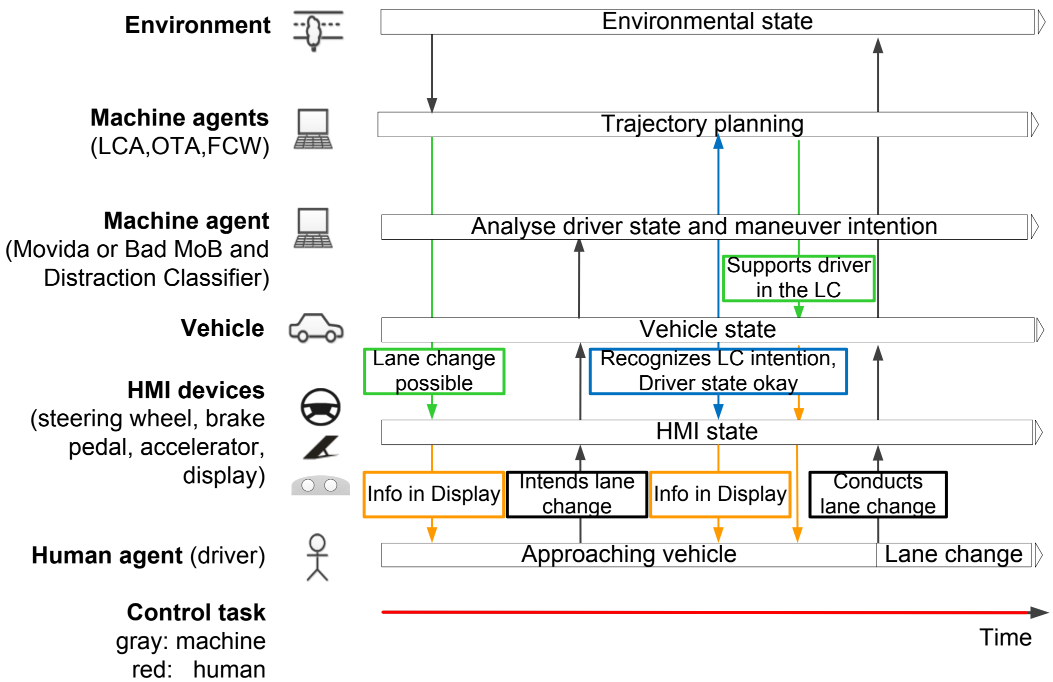

Figure 29: Sequence diagram for adapted assistance Scenario 1 "Normal situation – LC possible"

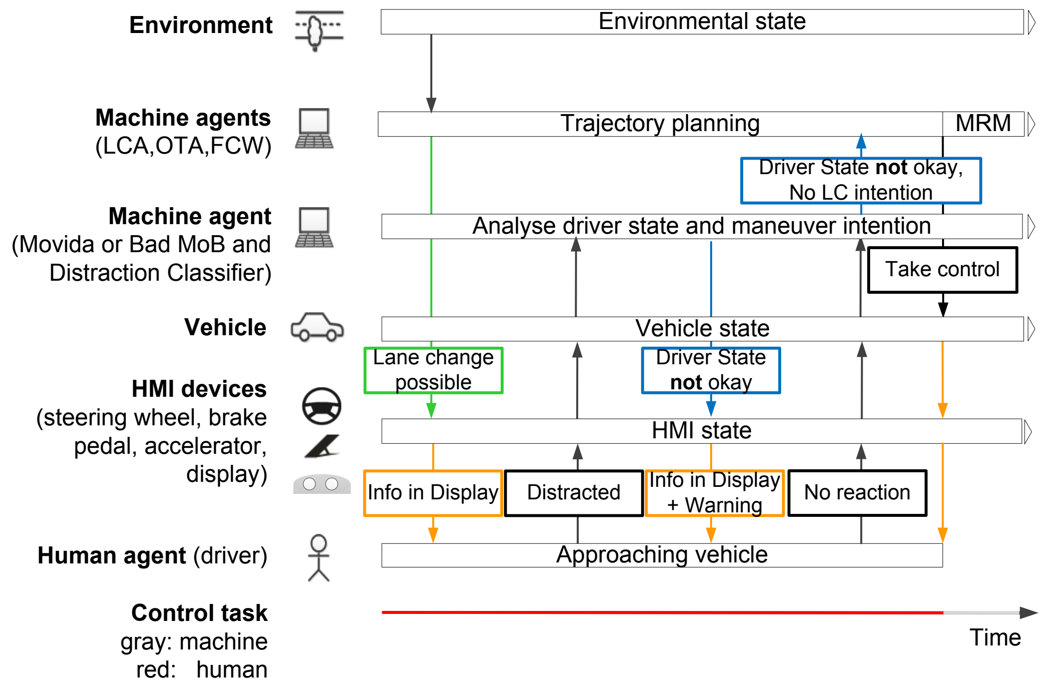

Figure 30: Sequence diagram for adapted assistance Scenario 2 "Driver state distracted"

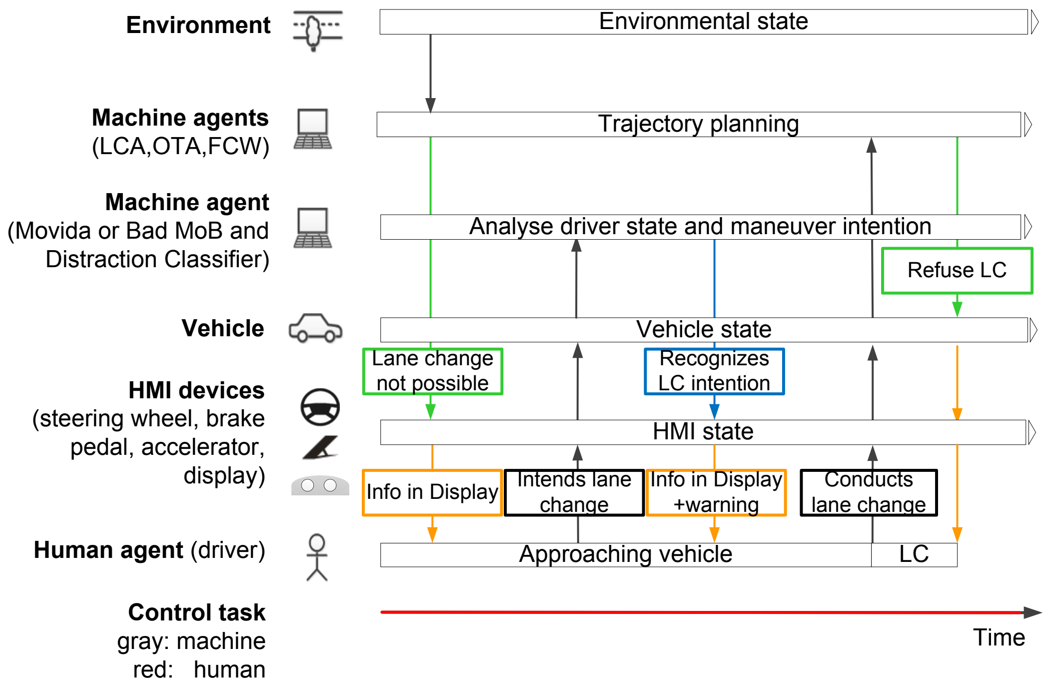

Figure 31: Sequence diagram for adapted assistance Scenario 3 "Refused LC"

Figure 32: Examples of communication elements for the adapted assistance use case (taken from Deliverable D.9.3)

Adapted Automation

Figure 33 and Figure 34 illustrate the sequence diagrams for the adapted automation use case. The two scenarios depend on each other since Scenario 1 illustrates the learning phase of the driver model. Its output is applied in the second scenario. As already mentioned in the adapted assistance use case the green boxes and arrows show the external context parts in the scenario. The external context relates again to information about

- recognized objects (surrounding traffic participants and traffic signs)
- the future path of the road including

- the current vehicle state like e.g. current velocity and acceleration
- the current state of the actuators like e.g. steering wheel, angles and pedal positions

The blue boxes address the internal context. These boxes indicate that for the internal context the WP3 tool CONFORM is used (see *Deliverable D3.3* and *D.9.3* for more details about CONFORM). CONFORM covers the context information about the driving style. CONFORM is used to adapt the driving style of the machine agents LCA, OTA and FCW (= automation style) at the end of Scenario 1. At that point the classified driving style of the current human driver is mapped to one of the pre implemented automation styles. Therefore CONFORM is responsible for the adaptation of the use case in Scenario 2. The communication in this use case is again visualized through orange boxes and arrows in the sequence diagrams. The communication of the adaptation happens currently only through indirect communication (action of the machine agent on the vehicle control and the movement of the vehicle and steering wheel). It is currently under discussion if we will communicate the applied automation style or not. For the communication of the external context an advanced HMI as described in *Deliverable D9.3* and shown in Figure 35 can be used.

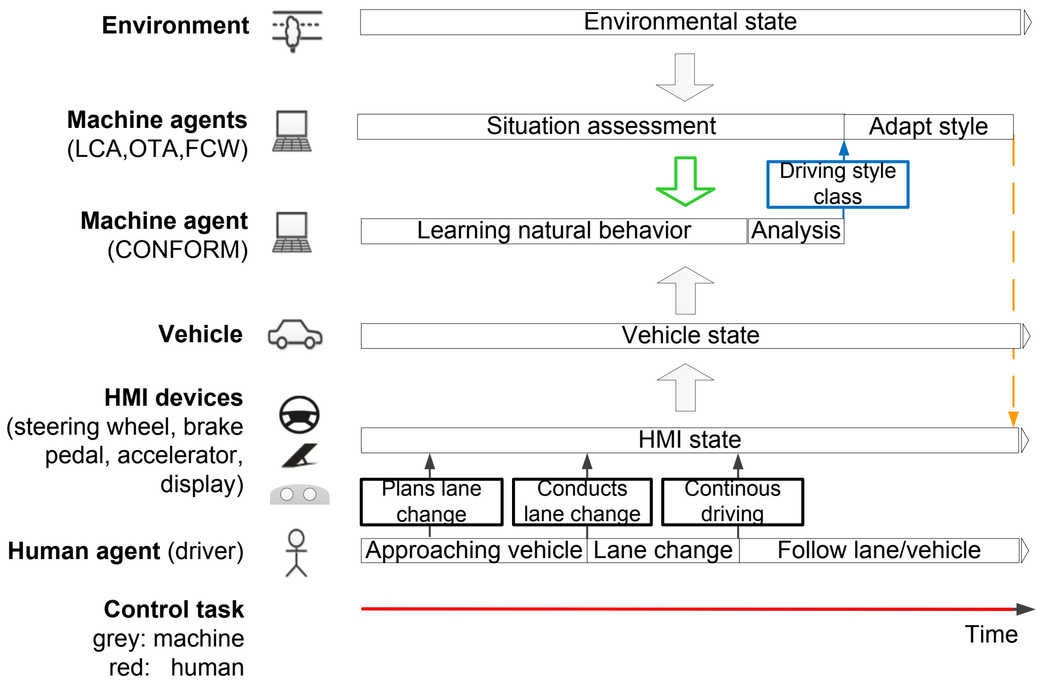

Figure 33: Sequence diagram for adapted automation Scenario 1 "Manual driving and adaptation of automation style"

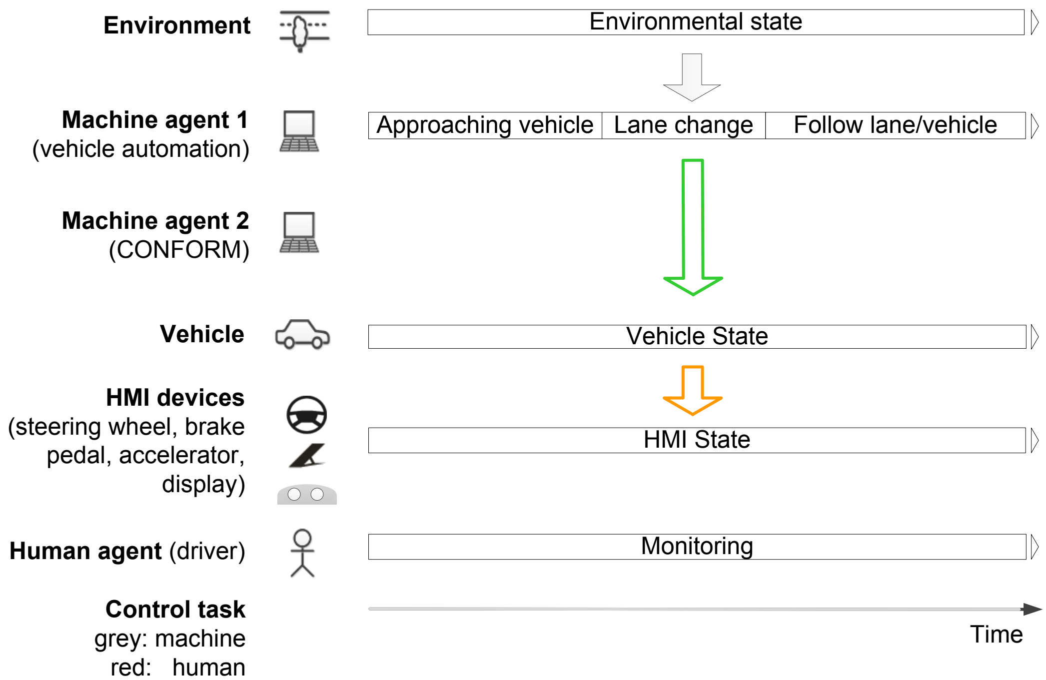

Figure 34: Sequence diagram for adapted automation Scenario 2 "Automated driving"

Figure 35: Examples of possible communication elements for the adapted automation use case (taken from Deliverable D.9.3)

Left display element - "Manual Mode" active and warning vehicle in blind spot.
Right display element – "Auto Mode" active and information about an automated lane change to the left.

4 Conclusion

We have presented many new concepts about our framework for adaptation in this deliverable. In the last deliverable we have introduced the key concept of the adaptation framework and proposed a graphical formalism to design AdCoS. In this deliverable we are going further and we describe the different element of our framework to implement and design such collaborative and adaptive systems. The main innovative idea of this framework resides in the two separate control loops: the executive loop and the adaptive loop. An agent closes an executive loop on a process and another agent (that could also be the same) closes an adaptive loop on the executive loop. Agents, machine and human, are then distributed, statically or dynamically to these loops, based on their respective competences/services, or factors such as workload, fatigue, vigilance for human agents. Then tasks and resources themselves are allocated to the agents, again statically or dynamically (adaptive task/resources distribution) based on the complexity of the process, tasks, resources and environment the AdCoS is dealing with. **Theoretically** our approach to formalize such system is not limited on the number of loops and it is possible to design very complex adaptive systems with other adaptive agent that could close adaptive loops on the adaptive loop.

This approach takes its root in control theory, cognitive modeling and the study of socio-technical systems. The relation with control theory is particularly obvious with simple AdCoS, with few control loops and few agents. For more complex AdCoS (with many loops/many agents), we are verging towards the field of complex socio-technical systems study and design, including the exciting new field of organic computing [17], which deals with such complex systems, with a peculiar focus on self-organization and adaptivity. In organic computing, controlled self-organization is seen as a way to maintain structure and functionality in the face of variable circumstances, internal or external to the system itself.

Such systems are characterized by their extreme flexibility, adaptability and resilience, and indeed belong to the realm covered by HoliDes, though the use cases we are dealing with in HoliDes resort from simpler systems.

In the second section of the deliverable, we have proposed a methodology to design AdCoS and described the different steps in detail. For now, we have explained how all elements of the framework should be linked together and we have suggested tools to model them. In the next deliverable, we should propose a formal model to design AdCoS using our adaptation framework philosophy. These models will be integrated directly in the HF-RTP. In parallel, we will implement and test some tools to manage some use case adaptivity needs (such as context assessment, adaptation computation and communication).

A new subsection has been developed for this adaptation framework, the HF guidelines. This guideline considers human factors before, during and after the implementation of adaptive systems and AA into a cooperative multi-agent-system (humans and machines). Besides definitions from the literature the guideline will provide step-by-step introductions on how to consider human factors in an appropriate way. In following deliverables, the guideline will be advancing in its development.

About the Use Cases specification, a new update and development iteration has been provided.

In the case of UC1, Guided Patient Positioning, the main advance has been on the Use Case characterization, describing the operation of the AdCoS. The details on AdCoS and environment, the operator actions and the different scenarios and their graphical communication to the operator have been presented.

For UC2, Diversion Airport, the updates from *Deliverable D3.4* in the definition of parts of the CoS – the pilot, the pilot monitoring system, the procedure monitoring system and the diversion assistant, the role of each part and their interactions have been elaborated. Regarding to the scenario detailed specification, the adaptation now uses new and updated tools as MED and pilot state classifier, evaluating their output data to perform the adaptation.

For UC3, Command and Control Room, the use case characterization has been extended by an explanation and a summary of the different UC chosen for this specific WP (UC3.1-UC3.5). Regarding the scenario detailed specification a description of physical and mental states covered by the system has been provided, together with the actions taken for the system to adapt to this states of the operator.

And finally, in the case of UC4, Overtaking including lane change, compared to *Deliverable D3.4* only a minor change has been performed, namely the addition of a forth machine agent: the driver model "DM". In the scenario detailed specification the information flow between the different agents within the use case is explained in more detail. The information flow is detailed by using the concept of sequence diagrams.

To conclude, we have proposed a methodology to design AdCoS and described the different steps in detail. For now, we have explained how all elements of the framework should be linked together and we have suggested tools to model them. In the next deliverable, we should propose a formal model to design AdCoS using our adaptation framework philosophy. These models will be integrated directly in the HF-RTP. In parallel, we will implement and test some tools to manage some use case adaptivity needs (such as context assessment, adaptation computation and communication).

5 References

[1] Parasuraman, R., Cosenzo, K. A., & De Visser, E. (2009). Adaptive automation for human supervision of multiple uninhabited vehicles: Effects on change detection, situation awareness, and mental workload. *Military Psychology*, *21*(2), 270.

[2] Byrne, E. & Parasuraman, R. (1996). Psychophysiology and adaptive automation. Cognitive Science Laboratory, The Catholic University of America, Washington, DC 20064, USA. Biological Psychology 42, p. 249-268.

[3] Draper, J. V. (1995). Teleoperators for advanced manufacturing: applications and human factors challenges, *The International Journal of Human Factors and Manufacturing*, 5, 53–85.

[4] Corso, G.M. and Moloney, M. M. (1996). Human performance, dynamic function allocation and transfer of training, in R. Koubek and W. Karwowski (eds), *Manufacturing agility and hybrid automation* (Amsterdam: Elsevier), 636–639.

[5] Wickens, C. D. (1992). *Engineering psychology and human performance* (2nd ed.). New York: HarperCollins.

[6] Parasuraman, R., Molloy, R., & Singh, I. L. (1993). Performance consequences of automation-induced 'complacency'. *The International Journal of Aviation Psychology*, *3*(1), 1-23.

[7] Calefato, C., Montanari, R. & Tesauri, F. (2008). The Adaptive Design, Human Computer Interaction: New Developments, Kikuo Asai (Ed.), ISBN: 978-953-7619-14-5, In Tech.

[8] Rusnock, C. F., & Geiger, C. D. (2013, September). The Impact of Adaptive Automation Invoking Thresholds on Cognitive Workload and Situational Awareness. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting (Vol. 57, No. 1, pp. 119-123). SAGE Publications.

[9] Scerbo, M. W. 1996, Theoretical perspectives on adaptive automation, in R. Parasuraman and M. Mouloua (eds), *Human performance in automated systems: Theory and applications* (Mahwah, NJ: Lawrence Erlbaum Associates), 37–64.

[10] Dorneich, M. C., Ververs, P. M., Mathan, S., Whitlow, S., & Hayes, C. C. (2012). Considering etiquette in the design of an adaptive system. *Journal of Cognitive Engineering and Decision Making.*

[11] Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *30*(3), 286-297.

[12] Inagaki, T. (2003). Adaptive automation: Sharing and trading of control. Handbook of cognitive task design, 8, 147-169.

[13] Lüdtke, A., Osterloh, J.-P., Mioch, T., Rister, F., Looije, R., (2009). Cognitive Modelling of Pilot Errors and Error Recovery in Flight Management Tasks. Proceedings of the 7th Int. Working Conference on Human Error, Safety, and System Development (HESSD 2009) Springer Verlag.

[14] Laird, J. E. (2008). Extending the Soar cognitive architecture. Proceedings of the Artificial General Intelligence Conference. Memphis, TN: IOS Press.

[15] Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. Psychological Review, 111, (4). 1036{1060.

[16] Langley, P., Cummings, K., & Shapiro, D. (2004). Hierarchical skills and cognitive architectures. Proceedings of the Twenty-Sixth Annual Conference of the Cognitive Science Society (pp. 779-784). Chicago, IL.

[17] Organic Computing — A Paradigm Shift for Complex Systems", see http://www.springer.com/us/book/9783034801294