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

D 3.3 - Framework for Adaptation

Project Number:	332933
Classification:	Public
Work Package(s):	WP3.2
Milestone:	M1
Document Version:	V 1.0
Issue Date:	14.08.2014
Document Timescale:	Project Start Date: October 1, 2013
Start of the Document:	Month 7
Final version due:	Month 11
Deliverable Overview:	**Main document:** D 3.3 - Framework for Adaptation, Public
Compiled by:	Richard Leblond - EAD-FR
Authors:	Nicolas Schneider, Richard Leblond, Denis Javaux, C. Delgrange (EAD-FR) Denis Martin, Cristobal Curio, Svenja Borchers (TWT) Peter van der Meulen (PHI) Mark Eilers (OFF) Nacho Gonzalez , Jordi Fonoll (ATOS) Thierry Bellet, Jean-Charles Bornard (IFS) Nicolas Dulac (INT) Simona Collina, Federica Protti (SNV) Lennart Weber, Franziska Michalczik(EAD-DE) Fabio Tango (CRF) Martin Boecker(CAS)
Reviewers:	Daniel Prun (ENA)
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
26.05.14	Initial Version	Richard Leblond (EAD-FR)
30.06.14	Role of TWT	Denis Martin, Cristobal Curio (TWT)
30.06.14	State of the art : dynamic task allocation	C. Delgrange (EAD-FR)
07.07.14	Chapter 2 added. Proposal for Chapter 2. Elements for Chapter 4	Denis Javaux (EAD-FR)
08.07.14	State of the art: General Frameworks	Svenja Borchers, Denis Martin (TWT)
08.07.14	New version of Chapter 2	Richard Leblond(EAD-FR)
10.07.14	New version of Chapter 7.4	Fabio Tango (CRF)
11.07.14	New version of Chapter 8	Nicolas Dulac (INTEMPORA)
16.07.14	New version of Chapter 8	Nacho Gonzalez(ATOS)
16.07.14	State of the art: previous work, new trends and general Frameworks	Jordi Fonoll(ATOS)
16.07.14	Chapter 7.1 added	Peter van der Meulen (PHI)
19.07.14	Rôle of IFS + Contribution to the Human Model Section (distinction between “Simulation Models” and “Models for Activity Analysis”)	Thierry Bellet, Jean-Charles Bornard (IFS)
19.07.14	New version of chapter 5	Simona Collina, Federica Protti (SNV)
21.07.14	New version of chapter 2	Lennart Weber, Franziska Michalczik(EAD-DE)
25.07.17	Chapter 7.3 added	Martin Boecker (CAS)

31.07.17	Chapter 6 added	Nicolas Schneider (EAD-FR)

Table of Contents

List of figures

List of tables

1 Introduction

This document serves the purpose of finding a common understanding of adaptation as well as the formulation and definition of terms all project partners can agree on. At first, the framework for adaptation is outlined theoretically, on the basis of which the project partners from different industries elaborate their AdCos and modeling approaches. This document is a living document that will evolve during the project.

2 Adaptive – Cooperative Systems

This chapter consists of a framework for adaptive-cooperative systems and their properties. The primary goal is the creation of an unambiguous terminology and semantic categorizations comprehensible for all project partners.

2.1 Human-Machine Systems: Automation

An essential part in Human Nature is the fact that humans pursue goals. Whatever a human does, be in searching for food or traveling from one place to another, it is goal-driven. Although humans are mentally and physically very capable in some respects, they suffer severe lacks in other respects. For this reason, machines were and are being invented in order to compensate weaknesses and facilitate life. The physical and cognitive capabilities of machines are steadily improving, allowing for a higher number and a higher level of abstraction of tasks solely accomplishable by machines. This development goes along with the question what tasks and what processes should be allocated to a machine, and when they should be allocated. On the other hand, questions regarding risk and operators' situation awareness arise. The allocation of tasks and functions (see section 2.1.2.1) to a machine is called automation.

2.1.1 Levels of automation

The automation of a task can be realized on different levels, ranging from no automation at all (the human is in charge of all cognitive and physical processes) to full automation (the system is in charge of all cognitive and physical processes, ignoring the operator). Levels of automation can be categorized using scales.

A well-known example is the scale introduced by Sheridan and Verplank (1978). It has been significantly used in human-machine interaction research, in particular by Parasuraman and his colleagues (e.g. Parasuraman & al, 2000 [42]; Miller & Parasuraman, 2007 [67]). The scale shows how a task can be shared between a human operator (human agent) and automation (machine agent), ranging from purely manual execution (level 1) to full automation (level 10).

Level of Automation	Description
1	The computer offers no assistance: human does it all
2	The computer offers a complete set of decision/action alternatives
3	The computer narrows the selection down to a few
4	The computer suggests one alternative
5	The computer executes that suggestion if the human approves
6	The computer allows the human a restricted time to veto before automatic execution
7	The computer executes automatically, then necessarily informs the human
8	The computer informs the human only if asked

9	The computer informs the human only if it, the computer, decides to
10	The computer decides everything, acts autonomously, ignoring the human

Table 1: Levels of automation (adapted from Parasuraman & al, 2000)

This taxonomy does not consider the level of the individual functions of a task, and how they can be shared between human and machine agents. Moreover, this scale mixes and superposes elements of task sharing (who does what), authority (who decides who does what) and human-machine communication (what the computer communicates to the human) that belong to different dimensions of the human-automation situation. A more elaborate account of task and function allocation is given in the following sections.

2.1.2 (Automated) Systems: Functions and Task Allocation

2.1.2.1 Tasks and functions

Humans and machines interact in order to execute **tasks**. These are characterized by a certain **goal** as well as steps of action and sub-processes that need to be carried out in order to be successful. These sub-processes are called **functions**. A function can be sensory, cognitive or behavioral nature. These processes are explained in the following.

2.1.2.2 Processes of functions

Essentially, functions can take on five different forms of processes:

- **Perception**: these are low-level sensory information reception processes. The perceiving agent has to decompose the input to separate components, detect movements or changes, and perceive elements and their properties (e.g. color, lightness, orientation, shape, speed etc.).

- **Evaluation**: the received information is evaluated in relation to current circumstances (e.g. normal situation, abnormal situation, a peculiar dysfunction...) and to current goals and priorities.
- **Decision-making**: once information has been evaluated, the agent(s) need to decide what action is required. The output of the decision-making process is an intention of action.
- **Action planning**: after having made a decision, the execution of the required action needs to be planned: these actions can be very simple but also at times require complex coordination of lower level actions in interaction with different agents.
- **Action implementation**: once the action plan has been produced, it is executed step by step.

Functions are often carried out in a stable and recursive temporal order (perception, evaluation, decision-making, action planning, and action implementation). This is called a cognitive control loop. Cognitive control loops have been proposed by many authors (e.g., Norman's Cognitive Loop , 1989 [68]; Wickens, 1992 [70]; Rassmussen's Decision Ladder (Vicente, 1990 [69]); Hollnagel's Small Model of Cognition, 1998 [66]). The Figure 1 illustrates a cognitive control loop.

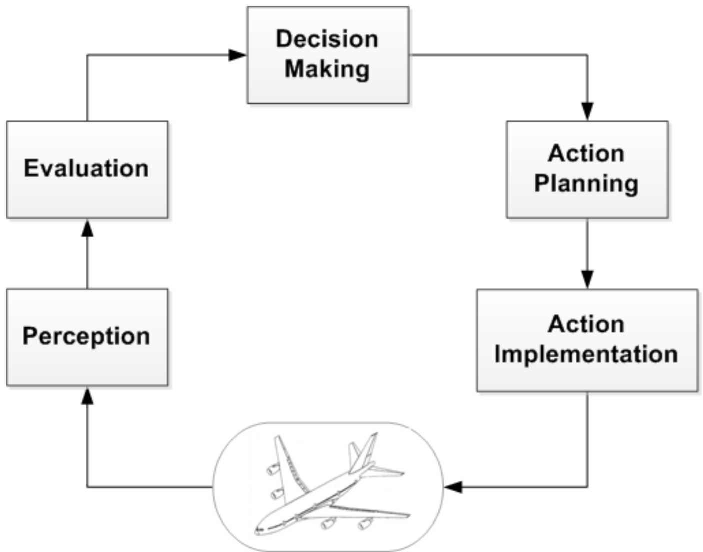

Figure 1: Cognitive control loop

Tasks and functions can be performed either manually (i.e., by a human agent only), automatically (i.e., by a machine agent only) or in a shared way (i.e., by both human and machine agents contribute to task execution).

The examples below show how the distinction between manual, automatic and shared processing applies to the five steps of a cognitive control loop:

- **perception**
  - *manual*: the human agent perceives the target object/plant/process naturally, through his or her natural senses
  - *automatic*: all information reception occurs through mechanical sensors. It is subsequently aggregated by a dedicated machine agent. Information is then presented to the user through some user interface for evaluation
  - *shared*: the human agent perceives some aspects of the current situation, being complemented by information obtained by a machine agent
- **evaluation**

  - *manual*: the human agent evaluates the information perceived autonomously
  - *automatic*: the information obtained from the perceptive stage (typically automatically, meaning it is fully available to machine agents) is evaluated by a machine agent
  - *shared*: the information obtained from the perceptive stage (either automatically or in a shared way) is evaluated partially by the human agent, with some assistance from a machine agent
- **decision-making**
  - *manual*: the human agent makes the decision autonomously. For example, after getting a warning about some system failure (automatic perception and evaluation), the human agent decides to call the maintenance team for on-site intervention (manual decision-making)
  - *automatic*: decisions are made automatically by a machine agent. For example, in commercial airliners, a malfunction can be detected automatically (automatic perception and evaluation) and command (decision-making) the display of a dedicated checklist (action planning)
  - *shared*: There are various ways of assisted decision making. For example, a machine agent could present only a subset of options to a human agent and let him or her make a decision among these options.
- **action planning**
  - *manual*: the human agent plans all actions, without any machine assistance. For example, in general aviation (small aircraft), the pilot plans how to fly and land at a given airport.
  - *automatic*: actions plans are generated automatically, by one or more dedicated machine agents. For example, in commercial airliners, checklists for addressing specific system failures are displayed (on the ECAM display in Airbus aircraft). The crew (human agents) will then have to execute the checklist at the action execution stage.
  - *shared*: assistance to the human agent for action planning can be provided through various ways: for example proposing tentative action plans the human agent has to approve.

- **action execution**
  - *manual*: the human agent executes an action manually. For example, controlling the trajectory of a car through the steering wheel is still achieved by the driver.
  - *automatic*: machine agents perform the actions. A comprehensive example is the autopilot in aviation.
  - *shared*: some of the actions are performed by the human agent and the others by one or more machine agents. For example, assistant systems in cars provide parking assistance, with the automation controlling the wheel and the driver the speed of the car.

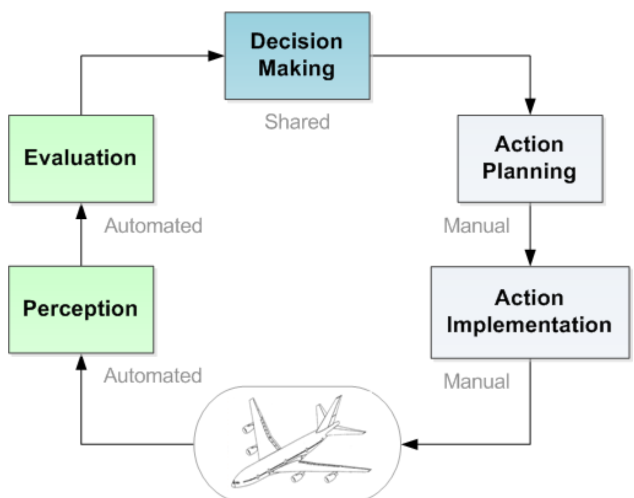

Figure 2: Assistance to a human agent in the performance of a control loop

Figure 2 shows an exemplary distribution of functions among human and machine agents (green = automatic, dark blue = shared (or mixed), light blue = manual): perception and evaluation of the situation are fully automatic (e.g. data aggregation, trends, warnings & alarms) meaning the human agent does not perceive the process directly but through dedicated user interfaces. The information provided to the human has a pre-evaluated form, supporting the human agents' decision-making. Decision-making is thus shared by the human agent and one or more machine agents, for example with machine agents suggesting several alternative decisions and the human agent making the final decisions. Action planning and action implementation are fully manual and thus performed by the human agent.

The types of processes elaborated above can be allocated according to different strategies. These are explicated in the next section.

2.2 Adaptation

This subchapter will provide a comprehensive overview of the general field of adaptation and the multiple facets it comprises. The first general definition of adaptation is elaborated in the respective subchapters. For a holistic, comprehensive understanding of the topic, the distinctive features of adaptation are absolutely essential.

2.2.1 Defining Adaptation

Since automated human-machine systems are highly complex in their structures and applications, they have to be dynamic to fit their purpose. They have to adapt. **Adaptation** is a response to environmental conditions: the process of adjustment to given (internal or external) conditions. It refers to any change in the structure or function of system that allows it to act more effectively, efficiently and successfully in its environment. Automated systems are usually designed to provide some form of adaptation on the level (and the type) of automation in order to deal with situational demands placed on the operators, human and machine. HoliDes is guided by four main questions concerning adaptation: What, Why, How and Who. These questions refer to specific aspects of adaptation: system design, adaptability and adaptivity. Figure 3 depicts the connections between those aspects. It is important to underline that every human-machine system can comprise all aspects. However, there are systems that are built on the principles of adaptability alone.

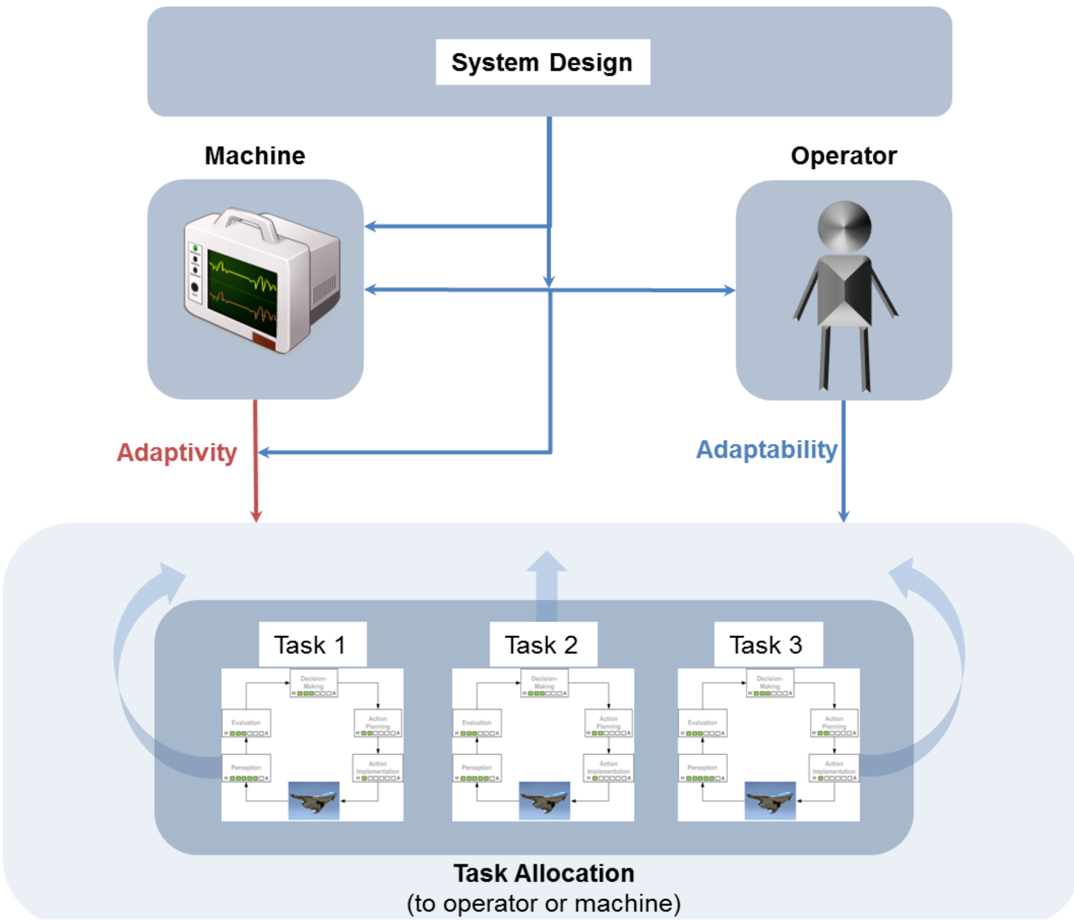

Figure 3: Human-Machine Systems from an adaptation view

- **What** is adapted: **System Design**. Tasks can be terminated or added, task and function allocation can be altered, new agents can be involved, information content and presentation, resources can be reallocated, added or withdrawn.

- **Why** should the system adapt: **System Design**. The "Why" is determined based on measuring, interpreting and/or predicting the internal context (e.g. the status of the human operators, or automated systems) and external context (e.g. changes in the number of objects under surveillance).

- **How** is adaptation performed: **Adaptability vs. Adaptivity**. The "How" defines the way in which adaptations are performed: simple predefined connections between internal and external context and respective reconfigurations, or more sophisticated dynamic optimization of tasks and resources. Main drivers behind the "How" are safety, robustness, resilience and efficiency.

- **Who** performs the adaptation: **Adaptability vs. Adaptivity**. Adaptation can be performed by one or more human operators, by one or more machine agents or by both (acting cooperatively).

2.2.2 System Design

All aspects of adaptive-cooperative human-machine systems have to be considered when analyzing or designing these systems. However, before dealing with the main differences between adaptability and adaptiveness of existing systems, the aspect of **system design** has to be considered.

When developing automated, thus human-machine systems of any kind, it is wise to consider system design aspects of adaptation. The foundation of it all is a thorough task analysis of the overall task and the respective subtasks that need to be carried out. By taking principles of adaptation into account, the development and testing of more elaborated systems can be far more cost-effective and time saving. This way the redesign after user tests is avoided and safety, efficiency and comfort/joy of use are increased.

When designing adaptive-cooperative systems, the issue of authority rises: "Who" has the authority to command adaptation. With adaptability, authority is in the hands of human agents. With adaptiveness, it is in the hands of machine agents. In practice, in many systems, adaptability and adaptiveness coexist and authority on adaptation is therefore shared between human and machine agents.

2.2.3 Adaptability

The authority over flexible control of information or performance in human-machine systems that are classified as adaptable, remains in the hands of the human. These system changes occur through one or multiple operators. The commands are static but yet occur mostly at runtime. Adaptability becomes necessary due to specific changes in the human-machine system, e.g. changes in information load resulting in changes in information types or task distribution.

Example:

a user ("Who") manually adjusts the luminosity of a display ("What") via a dedicated control ("How") to keep things readable ("Why")

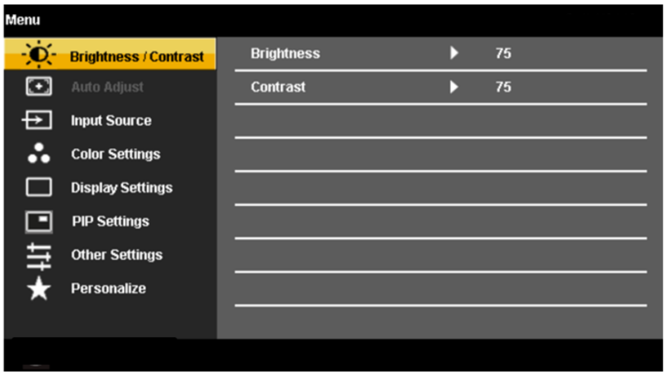

Figure 4: Manual adaptation of display luminosity (Dell)

One of the most advantageous characteristics of adaptable systems is the fact that the human operator is kept “in the loop”, meaning he is aware of the system’s status and the distribution of efforts on both sides, machine and himself and he is in charge of making decisions.

2.2.4 Adaptiveness

In adaptive systems the control of flexibility in information or performance alterations remains with the system itself, not the human. The System changes dynamically due to internal or external stimuli/events (environmental variables, operator demands and behavior).
Example:

> *A system ("Who") automatically activates ("What") or automatically adjusts the level of assistance it provides ("What")to maintain the human’s mental workload in an acceptable range ("Why")*

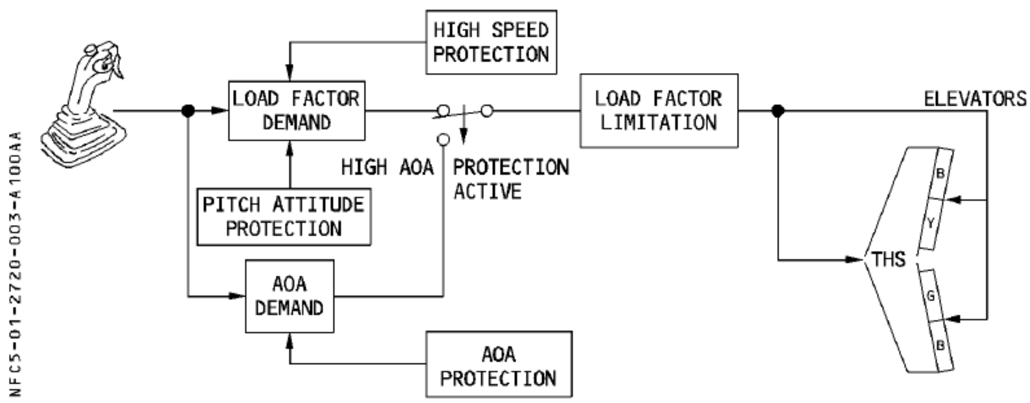

Figure 5 : Flight control protections (Airbus A320 family, from FCOM)

The **difference between adaptability and adaptiveness** is therefore mostly related to "Who" triggers adaptation: human operators (adaptability) or the system (adaptiveness). Adaptivity has certain advantages concerning speed of performance, reduction of mental workload and training time.

However, it has to be strongly advised to carefully weigh the opportunities and risks of both forms of adaptation in cooperative human machine systems.

2.2.5 Objects of adaptation

Both forms of adaptation concern three main objects which can be adapted. While the contents the systems are occupied with vary greatly from system to system and from industry to industry, there are some general objects of adaption from an abstract point of view: **tasks**, **task distribution** and **resource distribution**. Evidently, these objects are subject to different procedural changes depending on the type of adaptation they are faced with.

2.2.5.1 Tasks

In a human-machine system that is designed to be adaptable, the human operator would be able to change the nature of a superordinate task itself. The most common way to do so is to alter the subordinate tasks, e.g. to postpone certain activities, such as maintenance, or to assign a higher priority to them. An adaptive system would either reassign task timing or priorities automatically, or inform the human operator about the processes - depending on the level of automation.

2.2.5.2 Task distribution

Apart from the nature of the tasks, adaptable and adaptive systems organize the assignment of tasks to the different agents (system or human operator) differently. Tasks are typically redistributed because changes in the context occur. These context changes can be external (e.g., environmental conditions, another car comes on a collision course (automotive domain), an engine fire is detected) or internal (e.g. the operator’s capabilities change, machine agents fail, sensor data becomes unavailable).

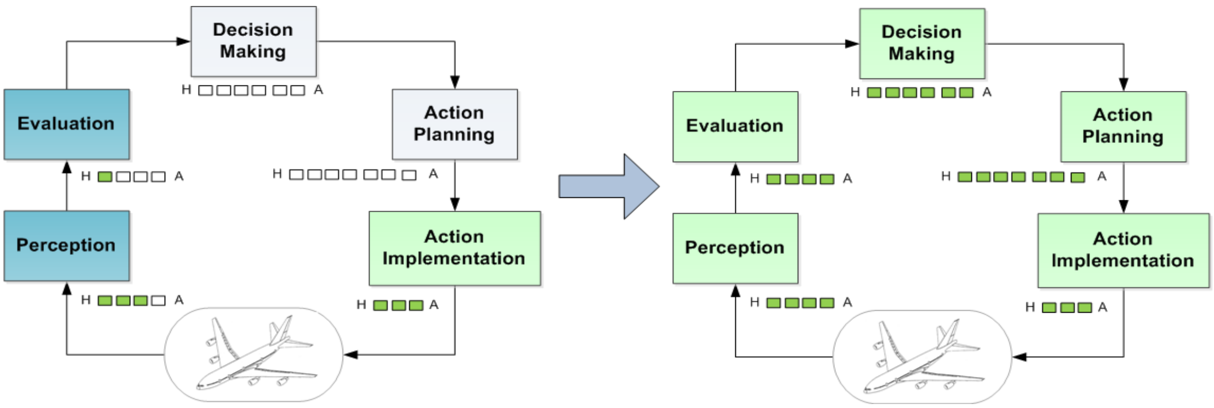

Figure 6 : Adaptation of task distribution

In figure 8 task distribution is adapted in an evolving situation. While the human agent is first involved in all steps of the cognitive control loop, with assistance from machine agents (especially at the perceptive and evaluative stages), the machine agents (automation) later fully close the loop and achieve all stages autonomously. This put the human agent in a position of supervisory control – a position where human operators typically do not show high performance.

2.2.5.3 Allocation of functions and tasks

The following strategies have emerged to allocate tasks and their subparts functions (see section 2.1.2), ranging from static to dynamic ones:

- MABA-MABA list ("men-are-best-at-machines-are-best-at"): This is the oldest and most static allocation strategy. In this list, certain abilities of humans and machines are juxtaposed. Based on which agent is best at what, functions are distributed.
- Leftover: This strategy consists of the automation of as many functions as possible. It is also a static strategy and entails a high risk of out-of-the-loop problems.
- Economic: This is a dynamic function allocation strategy: Based on an adequate assessment of situational system parameters, functions are

allocated to agents with sufficient abilities and current capabilities, resulting in an efficient way of resource consumption.

2.2.5.3.1 Strategies of adaptive function and task allocation (critical events, measurement-based, modeling-based)

When a machine agent dynamically adapts itself to external and internal situation changes, this can be done according to three different strategies, which are elaborated below:

- Adaptivity based on critical incidents: When machine agent(s) detect events that are critical to task completion or endanger the overall functioning of the system, they allocate tasks and function, as well as what information to provide to human agent(s).
- Measurement-based adaptivity: The machine agent(s) continuously measure the state of the environment as well as the behavior of human agents as well as other machine agents. For example, if an operator starts making minor errors at the end of a night shift, this is interpreted by the machine agent(s) as an indicator of fatigue or increased workload, and thus leads to appropriate countermeasures.
- Model-based adaptivity: Based on cognitive human and machine models, the machine agent(s) assess whether the current situation is likely to overburden the human agent(s).

Figure 7 illustrates these three adaptive strategies of task and function allocation.

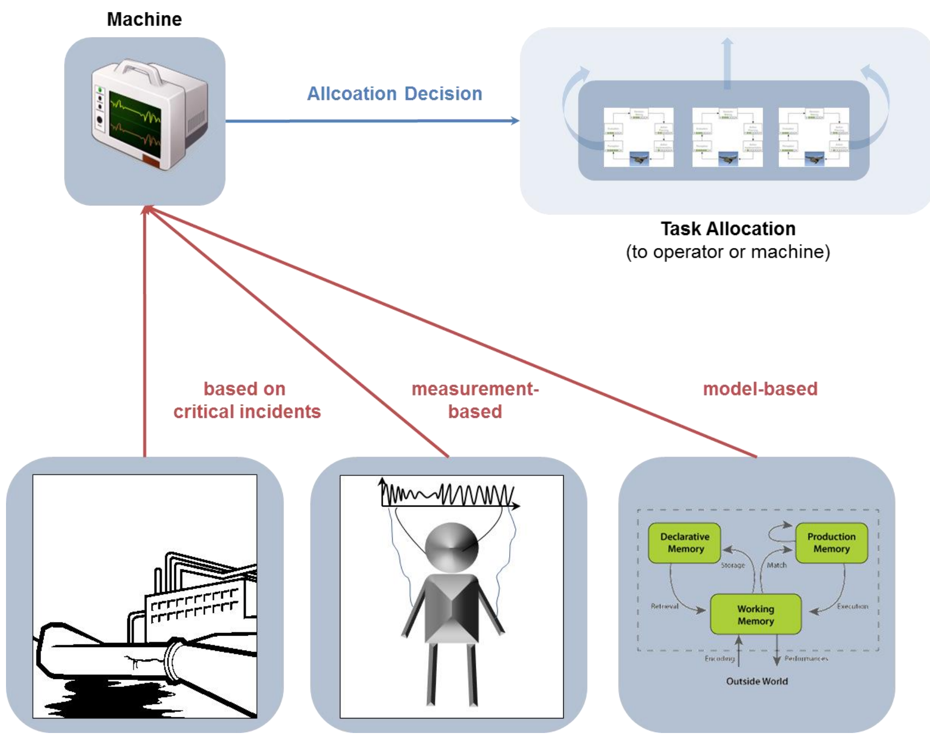

Figure 7: Adaptive strategies of task and function allocation

2.2.5.4 Dangers of task distribution in adaptable and adaptive systems

Some of the disadvantages of the supervisory control position are a **reduction of situation and system awareness**. This is especially the case in adaptive systems, where the machine agent takes decisions about task distributions, thus also about the distribution of information gathering and processing.

The adaptive redistribution of tasks can also lead to an increase in **mode errors**. These errors occur when the human operator is sharing task responsibilities with a machine agent and is taking action appropriate for one system mode, but the system is actually in another mode. An example in the healthcare domain would be the supervision of a complete station where a nurse want to mute a (false) asystolia alarm intended for one patient's monitor, but is actually muting alarms for the whole station – leading to severe consequences for other patients. The nurse was thinking the system is carrying out commands for a single patient when it's actually in master mode for all monitors.

Another pitfall of task distribution in adaptable and mostly in adaptive human-machine systems is the issue of **trust in human-machine systems**. Human operators tend to show inadequate complacency towards automated systems, especially in situations where they're put in a supervisory control position, thus mostly in adaptive systems when they're more likely to be out-of-the-loop. That means that human operators are less likely to question decisions made by the machine agent, even though there is evidence that the system might have failed. The higher the reliability of such a system, the stronger is the complacency.

When working in adaptive-cooperative system, there's always a risk of **skill degradation** for the human agents. This is especially the case when humans remain in a supervisory position and the machine agent makes all the decisions and carries out the work.

When adaptable and adaptive systems are configured to either extreme of task distribution, fully automatic or fully manual work settings, the consequence is very often an **unbalanced mental workload** in the human operator. Thus, if an operator is completely out-of-the-loop, his mental workload might be excessive when he suddenly has to regain manual control over a system.
All of the above points are responsible of a potential **performance degradation** of the whole human-machine system. It is wise to design systems in such a way that neither extreme of task distribution is carried out by the system (adaptivity) or the human (adaptability). This also mitigates another danger of task distribution in human-machine systems: **decreased user acceptance**. In complex adaptive-cooperative systems, the highly trained and skilled human operator wants to retain control. System designers

have to take this fact into account when making a human-machine system adaptable or adaptive.

Resource distribution: how the human and machine agents use and consume the resources is changed. This may happen because some resources become unavailable (e.g. sensors stop providing data, resources are used up) or the cost of using them increases and other resources have to be used (e.g. missing data is obtained or inferred from other sensors). Resource distribution also has to deal with mutual exclusion issues (some resources can only be used by a limited number of agents at a time), which may lead to their permanent adaptation.

2.2.6 Types of adaptation

While the aforementioned objects of adaptation answer the question "what" is adapted, the question "how" adaptation occurs is answered by the types of adaptation.

2.2.6.1 Temporal adaptation of interactions

When a human agent interacts with a machine agent, the interaction typically follows specific temporal structures. For example, the machine agent provides a warning on something, the human agent requests additional details on the warning, evaluates the situation and then decides to acknowledge the problem. This is an interaction pattern, which defines how information flows between the human and machine agent, in a specific temporal order. It is usually useful to design the interaction to resort to different patterns when the circumstances change. In case of high temporal pressure, the human agent must act fast and may bypass the steps where additional details are requested and then provided.

2.2.6.2 Adaptation of interaction modalities

Different modalities (e.g. vision, audition, touch, and proprioception) can be used to support the interaction between the human and the machine agents.

Frequently interaction is even multi-modal. Adaptation can occur by changing the modalities that define the interaction. For example, vision and audition (e.g., to provide auditory warnings) are used to support the interaction, but the noise level in the environment increases and auditory communication is no more an option. The same information is now communicated through enhanced reliance on vision (more salient visual warnings) and through the haptic modality.

2.2.6.3 Adaptation of user interfaces

The content of the human-machine interface consists of many aspects, the most prominent ones being:

- **Information presented to the user**: the information being presented may change, e.g. to begin another phase of operation, where different information is needed. How that information is presented (e.g. shapes, colors, types of widgets,...) can also be changed to adapt to the type of cognitive task the user is performing, to particular circumstances (e.g. increased salience of some information in abnormal situations), or to particular types of users (e.g. older users, use of larger fonts).

- **Controls made available to the user**: user interfaces are also used as an input device. The nature and functioning of the controls (e.g. menus, touch pads) may change to adapt to particular tasks or circumstances or certain user types. For example, larger touch pads could be used for older users.

2.2.7 Adaptation and communication

The system design and implementation of adaptation in the processes the human-machine system incorporates is evidently an essential part of the adaptive-cooperative human-machine system. However, one equally important part is the interface to the human operator, since he needs to be informed about system status in order to develop an appropriate degree of situation awareness and trust.

What is communicated through which communication means is strongly dependent on the object of adaptation, as well as the type of adaptation. Some adaptations are safety critical and automatic adaptations to e.g. route settings in an aircraft need to be communicated clearly and timely. In some other cases however, the human agents in the AdCos may not even need to be aware of the adaptation. For example, several machine agents cooperatively in charge of a task may decide to reorganize themselves, without any impact on the performance of the task. For the human agents, this does not matter, provided the set of machine agents still performs the task. This is frequently the case with automated systems that use a big foundation of e.g. computational networks.

Furthermore, in some cases adaptation does not need to be communicated explicitly, because **it is observable**. For example, when a human-machine interface automatically adapts its brightness to the current environmental context, the adaptation does not need to be communicated to the human user. The change is observable.

3 State of the art on adaptive systems

3.1 Previous research work in adaptive systems

There are 3 different steps:

The first step main conclusions is "what a self-adaptive system is and need"
In the second step: is based in self-configurations and web based server-client systems.
In the third step: Information is needed for decision making. Decisions need continuous revision due to new possibilities or problems.

This is the previous research work time line:

Software Architecture as a Design-time tool for Systems that Need to be Adaptive:

This step is divided in four parts:

1. Using Waves for software Construction and Analysis:[52]

- Architectural Style with accompanying notation.
- Tool fragments process object streams (NO byte streams).
- Connectors are explicitly sized queues.
- Tool fragments can have multiple inputs and outputs.

2. A Component and Message-Based-Architectural Style for GUI Software [53]:

- Obtain the benefits of MVC in a distributed and heterogeneous setting.
- Layered network of concurrent components hooked together by explicit message-based connectors.

3. Specific Distributed Software Architecture and a Constructive Development Environment for Distributed Programs [54]:

- Based on the notion of provided and required interfaces.
- Allows for Dynamic Configuration meaning the system's structure can change over time.

4. An Architecture-Based Approach to Self-Adaptive Software [55]:

- UAVs are used to disable an enemy airfield.
- Components are added to fielded and heterogeneous systems with no downtime.

Software Architecture for Self-Adaptive Systems:

In the second step:

1. The K-Component Architecture Meta-Model for Self-Adaptive Systems [56].

2. Dynamic Software reconfiguration in Software product families [57]:

- A Software Architecture that characterizes the similarities and variations that are allowed among the members of a product "family".
- Process of adapting the architecture of the product family to create the architecture of a specific product member.

3. Increasing System Dependability through Architecture-based Self-repair [58]:

- Provides a generalization of architecture-based self-adaptation.
- The architectural style becomes a First-class run-time entity.

4. Exploiting Architectural Prescriptions for Self-Managing, Self-Adaptive Systems: A Position Paper [59]:

- The goal is to minimize the degree of explicit management necessary for construction and subsequent evolution whilst preserving the architectural properties implied by its specification

5. Self-Managed Systems: An Architectural Challenge [60]:

- Architecture provides the required level abstraction and generality to deal with Self-management.

Ongoing Research on Software Architecture for Self-Adaptive Systems

The third step is:

1. An Architectural Strategy for Self-Adapting Systems and Pattern of Delegate Multi-Agent Systems [61]:
- Solutions: Smart messages mix state and behavior.
- About effectively using a conglomerate of smart messages to solve the problem of repeated interactions.
2. Using Architectural Models to Manage and Visualize Runtime Adaptation[62]:
- The main result is a historical graph of architectural configurations.
3. A-3: an Architectural Style for Coordinating Distributed Components [63].
4. A Languagefor Architecture-Based Self-Adaptation [64]:
- A language for defining and automating the execution of adaptation strategies in an architecture-based self-adaptation framework.

3.2 Related projects

Existing projects like D3CoS [30], HAVE-IT [31], CAMMI [32], ASTUTE [33] provide a techno-logical basis to build Cooperative Human-Machine Systems by providing and algorithms for measuring human operator and contextual states - these are first steps towards adaptive systems.

D3Cos [30]

Within the methodology of the D3CoS project, the steps of

- D3CoS Composition,
- D3CoS Interaction and,
- D3CoS Interfaces,

have been defined to support developers designing and evaluating a D3CoS system architecture in a structured way. An evaluation process should run in parallel to assess system and human operator performance, system robustness and system efficiency and network and communication efficiency.

HAVEit [31]

HAVEit aims at the realization of the long-term vision of highly automated driving for intelligent transport. The project develops, validates and demonstrates important intermediate steps towards highly automated driving.

Design of the task repartition between the driver and co-driving system (ADAS) in the joint system. It is of the most importance to ensure that the driver is in the loop when required.
It has to be ensured that he or she is able to react properly in a potentially critical situation.

CAMMI [32]

CAMMI provides innovative solutions for intelligent multi-modal interactive systems:

- Cognitive Monitor
  To monitor human cognitive state through operator and performance data acquisition and data processing, in order to optimize MMI interactions through workload mitigation methods.
- Workload Mitigator
  To assess and manage the measured cognitive state in order to understand any mismatch between the operator's current workload and the operational situation and to select the correct automatic MMI adaptation strategy.
- Adaptive MMI
  Implementation of workload-related adaptive strategies in order to trigger levels of automation assistance in multiple task and critical situations.

ASTUTE [33]

It's an advanced and innovative pro-active HMI interface and reasoning engine system for improving the way the human being deals with complex and huge information quantities, during real operations that without any type of assistance would saturate his performance and decision-making capabilities in different operative conditions and contexts.

3.3 New Trends in Adaptive Systems

The objective of this chapter is to provide some examples of the new and future trends in adaptive systems.

3.3.1 A Cooperative Predictive Control Approach to Improve the Reconfiguration Stability of Adaptive Distributed Parallel Applications

Adaptiveness in distributed parallel applications is a key feature to provide satisfactory performance results in the face of unexpected events such as workload variations and time-varying user requirements.

The adaptation process is based on the ability to change specific characteristics of parallel components (e.g., their parallelism degree) and to guarantee that such modifications of the application configuration are effective and durable. Reconfigurations often incur a cost on the execution (a performance overhead and/or an economic cost). For this reason advanced adaptation strategies have become of paramount importance.

Effective strategies must achieve properties like control optimality (making decisions that optimize the global application QoS), reconfiguration stability expressed in terms of the average time between consecutive reconfigurations of the same component, and optimizing the reconfiguration amplitude (number of allocated/de-allocated resources).

To control such parameters, TAAS (ACM Transactions on Autonomous and Adaptive Systems) propose a method based on a Cooperative Model-based Predictive Control approach in which application controllers cooperate to make optimal reconfigurations and taking account of the durability and amplitude of their control decisions.

The effectiveness and the feasibility of the methodology is demonstrated through experiments performed in a simulation environment and by comparing it with other existing techniques. [27]

3.3.2 A Learning-Based Framework for Engineering Feature-Oriented Self-Adaptive Software Systems

IEEE Members Mrs. Esfahani, Elkhodary and Malek [28] present an approach for engineering self-adaptive software systems that brings about two innovations:

- A feature-oriented approach for representing engineers' knowledge of adaptation choices that are deemed practical,
- An online learning-based approach for assessing and reasoning about adaptation decisions that does not require an explicit representation of the internal structure of the managed software system.

IEEE Members present an empirical evaluation of the framework using a real world self-adaptive software system. Results demonstrate the framework's ability to accurately learn the changing dynamics of the system, while achieving efficient analysis and adaptation.

The result of this research has been a framework, entitled FeatUre-oriented Self-adaptatION (FUSION), which combines feature-models with online machine learning. Domain expert's knowledge, represented in feature-models, adds structure to online learning, which in turn improves the accuracy and efficiency of adaptation decisions.

3.4 Adaptation Frameworks

In this section, general existing frameworks for adaptation are presented. The Collect-Analyze-Decide-Act and Sense-Plan-Act paradigms describe commonly used phases and components for adaptive systems (compare also to section 2.1.2.2).
In addition, the Observer/Controller and the Operator-Controller Module are both examples of architectures that focus on learned behavior and on-going optimizations.

3.4.1 Collect-Analyze-Decide-Act

Autonomic systems form a feedback loop, collecting information from several sources, analyzing them, forming a decision based on the analysis and reporting this result to users or acting in a similar way.

This process is also often referred to as the autonomic control loop [4]. Specifically, in the collection phase relevant knowledge information about the current state is collected, e.g., via environmental sensors or network instrumentations. This data must be analyzed in the next step constructing a model of the situation using inferences and distinct rules. At this state it needs to be clarified how the systems state is inferred and which data is relevant for validation.

The basis of the inferences is an useful knowledge for the decision making process in the next step. In the acting phase, the decision is attempted to be realized by performing the adaptation or by reporting the result to users or administrators. For the next control cycle, the impact of the decisions can be fed back and used as relevant knowledge.

3.4.2 Sense-Plan-Act

The sense-plan-act (SPA) is another approach for autonomous robots consisting of three functional components: a sensing system translating raw sensor inputs into a world model, a planning system generating a plan to achieve a specific goal with the help of the world model, and an execution system generating the actions provided by the plan [12].

The characteristics of the SPA approach are that the flow of control among these elements is unidirectional and linear and that the acting component, i.e., the execution of a plan, is built of orderings, conditionals and loops. Thus, the intelligence of the system is entailed in the planning component that generates the plan.

However, the SPA architecture entails the major difficulty that planning is time-consuming. Since the world may change quickly, the resulting plan might be rendered invalid already during the planning process. Thus, these time-consuming computations induce the risk of internal states that are not synchronized with the reality that it is intended to represent and therefore execution steps might be executed in an inappropriate context [12].

3.4.3 Observer/Controller in Organic Computing

Numerous sensors, processors and embedded systems provide safety and comfort functions as well as regulation or motor control functions. These embedded systems will be interconnected and form a complex communication network.

A system consisting of many interacting components may exhibit new properties emerging from new configuration possibilities that are not yet anticipated in the design stage but need to be dealt with at run-time. This requires adaptive systems with optimization techniques in order to learn adequate responses to unforeseen conditions.

A generic observer/controller is required to assess the behavior of an organic computing (OC) system and to control its dynamics [19]. A number of sensors and actuators are used in order to measure system variables and to influence the System under Observation and Control (SuOC) characterizing the system's global state and dynamics.

The observer measures and quantifies the current state of the SuOC. The monitored data needs to be preprocessed, analyzed and a prediction of future developments will result in situation parameters that characterize the observed or future system state.

Based on these situation parameters that are computed by the observer and being transferred to the controller, an evaluation will be performed with respect to the user-defined goal leading to a decision of the controller whether an intervention will be beneficial.

This decision is made by mapping the situation parameters to respective actions and evaluating the performance changes. Previous situation-action mappings will be stored in order to determine the reaction to known situations. Using these mappings and estimations, the controller will basically act as a learning component.

In particular, the controller is designed in two levels consisting of an on-line learning level and an offline optimization level. This design provides several advantages: using simulation based evaluations, appropriate situation-action mappings can be found without having to test different alternatives and this approach is significantly faster than the realization of evaluations in the SuOC.

Combining the slower level 2 approach with the faster memory-based level 1 approach enables a quick reaction by situation-action mappings for known or similarly known situations while in parallel an optimized situation-action mapping will be available in the future [21]. Therefore, the observer/controller architecture framework is widely applicable to a large range of technical systems.

3.4.4 Operator-Controller Module with Learning Procedures for Optimization

Another related approach is the Operator-Controller Module (OCM) developed by the Collaborative Research Centre 614[2]. The OCM is an autonomic system following its own objectives. It is specialized for mechatronic systems combining mechanical and electrical engineering with a strong focus on real-time constraints [20].

The controller represents the continuous part of the system and the operator comprises the time-discrete parts of information processing, which includes functions like emergency routines, controller monitoring and optimization. In particular, a reflective operator may modify the controller and induce switches between control strategies, while a cognitive operator gathers information about the system and its environment improving the system's behavior.

The agent could, e.g., use simulation runs of alternative future behaviors and evaluate them selecting the most promising alternative concerning the optimization goal.

Thus, the structure of the OCM is especially useful for model-based optimization and due to its modular composition it is easily possible to add other methods or functions of the agent theory. During execution of the plan

---

[2] Universität Paderborn. Collaborative Research Centre 614 website: http://www.sfb614.de/

and the monitoring of the real world, inductive and reinforcement learning is used in order to adapt the behavior-based environment and system models to the real world.

The use of learning procedures enhances the assessment of an optimal starting point for the optimization and the convergence of the optimization technique. Hence, this knowledge base can be used for similarity analysis enabling the detection of frequently reoccurring scenarios [20].

3.5 Dynamic Task Allocation

3.5.1 Dynamic Task Allocation: An adaptive mechanism

Adaptation is viewed in [22] as closed-loop mechanism. Process starts by monitoring environment, software entities (ie: memory, CPU load) or human operator (ie: neurophysiological and physiological sensing) to generate a collection of data. On the other side, effectors act exclusively on the monitored system by applying the selected adaptation action.

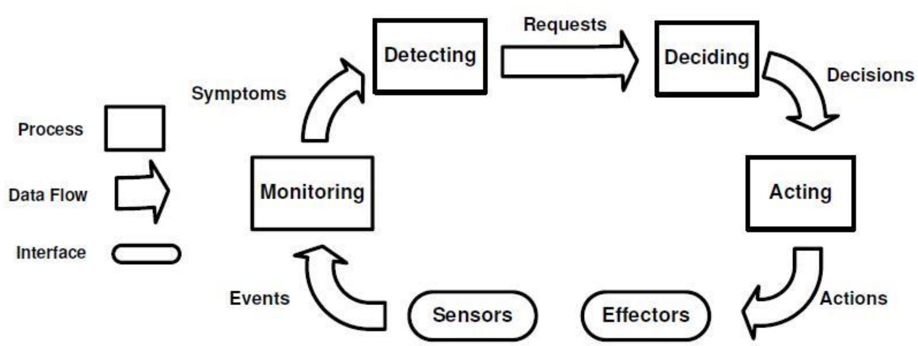

Figure 8: Four Adaptation Processes in Self-Adaptive Software.

The four processes shown in Figure 8 are described below from the standpoint of a dynamic task allocation:

Monitoring:

The monitoring process realizes a situation awareness and a context assessment. Situation awareness means that the system is aware of its self-state, it is described in [8].Context-aware provides an understanding of its operational environment, it is discussed in [1] and Bolchini realizes a survey of context models in [2]. One can re-call techniques employed in context assessment: logic-based, ontology-based or classical database systems. To permit a task reallocation, it could be useful to correlate the cognitive load of the operator data and how a task goes from its pre-condition to its post-condition over the time.

Detecting:

The detecting process analyzes the symptoms provided by the monitoring process to decide when a task should be reallocated. Accordingly to the taxonomy of triggers provided by Feigh in [10] for adaptation represented in Figure 9, operator could be allowed to decide to leave a task or in more adaptive manner, a new allocation could be triggered as follow:

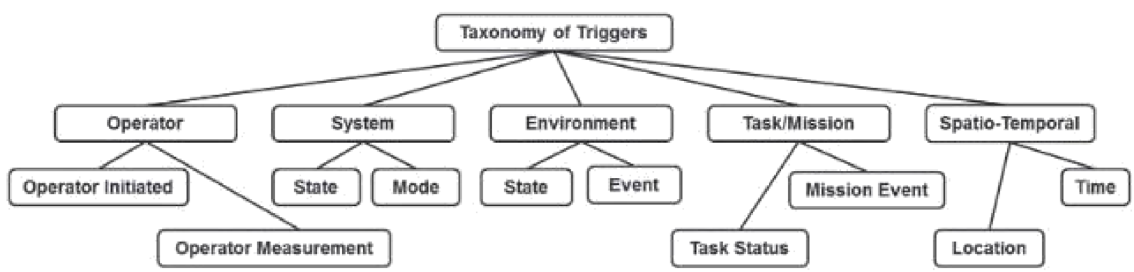

Figure 9 : Taxonomy of Triggers for adaptive systems

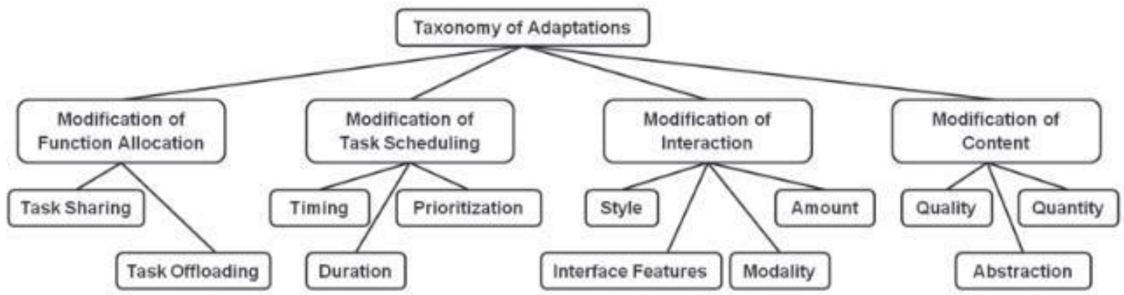

Figure 10: Taxonomy of Adaptations for adaptive systems

Deciding:

The deciding process selects which adaptation type to apply. In our case, only task allocation is considered but other adaptation types may be found in [10] and show in Figure 10. How to apply the adaptation is another issue and deals with proactive behavior and how avoid future disturbances.

Acting:

The acting process applies the actions determined in the deciding process. For instance, this could be charging or discharging an activity and provide feedback to the operator or call a new specific service.

The HoliDes' point of view of adaptation is dynamic only. Thus, this document puts aside adaptation mechanism from design phase of software engineering processes. However, how to fulfil Non-Functional Requirement at all-time introduced in [24] or a Goal-Oriented design presented in [18] may be relevant.

3.5.2 Why is an agent-based solution appropriate?

Wooldridge provides in [25] four factors to consider in order determining if an Agent-Based solution is appropriate. The HoliDes' assumptions are analyzed under these factors below:

Agents are a natural metaphor: HoliDes describes Cooperative Human-Machine systems as a set of agents who cooperate to achieve a common goal. Since the adaptation process occurs at a local level, it should be modelled in the same way that the system on which it acts.

Distribution of data, control or expertise: Systems in HoliDes encompass several operators or systems with different skills such as a pilot and an automatic system which diagnostic an electrical emergency and where each agent accesses to specific resources. Thus, data, control and expertise are inherently distributed.

Highly dynamic environment: HoliDes deals with realistic industrial problems from aeronautic, automotive, control rooms or health domain. In this context, maintain safety at run time must hold in a highly dynamic environment.

Legacy: It is important to note the difference between the system to adapt and the adaptation process. A component of a system in HoliDes might be technologically obsolete from the point of view of communicative and cooperative skills, but functionally essential. An Agent Level on top of each component of the system might be the only way to permit adaptation of the overall system.

Many Agent-Based modeling software may be considered to develop an Agent-Based solution, most of them are provided in the latter section.

3.5.3 Dynamic Task Allocation : An optimization problem

Dynamic task allocation is seen by Zlot and Stentz in [26] as the combination of two deciding processes: “What do we do?” and “Who does what?” The former deals with practical reasoning and involves planning and scheduling, while the second involves social decision making including several concepts such as utilities and preferences, game theory or bargaining.

Making a relevant task allocation is somehow an optimization problem where the optimization criterion depends of the meaning of relevant. For instance, a system could wish to improve time or load balancing between each agent, or, in a more complicated case, a set of criteria such as time, load balancing and safety. Compare task allocation on a set of criteria involves the use of a utility function. Gerkey and Mataric in [11] provide a formal analysis of utility as an expected quality of task execution and an expected resource cost.

Many studies have been carried out on the topic of task allocation with several optimization criteria and different assumptions on tasks, agent skills and the environment. [11] provides a taxonomy of task allocation in Multi-Robot system. Single-task robots, single-robot tasks and instantaneous assignment is considered as the simplest task allocation problem, while multi-task robots, multi-robot tasks and time-extended assignment the hardest. HoliDes considers task dependency and dynamicity of task allocation too, and the multi-robot task allocation problem could be characterized as follow:

- Each agent is multi-task (multi-task robots).
- Tasks might be performed by several agents (multi-robot tasks).
- A model of how tasks are expected to arrive exists (time-extended assignment).
- Tasks are interdependent (interrelated utilities and task constraints).
- Dynamic environment (On-line task allocation algorithm).
- Optimization: Minimize gap with the nominal case.

Accordingly to [11] this problem is strongly NP-Hard and no formal heuristic exists, however several approaches has been proposed and validated in a proof-of-concept fashion. In [26] a decentralized approach based on task trees and a market-based task allocation is given, nevertheless task interdependency does not include inter-related utilities and only AND/OR constraints are permitted. In [13] a similar approach is given, the solution interleaved the "What do we do?" and the "Who does what?" in decentralized manner with I-Plan, an HTN planner, the Contract Net Protocol and social commitments.

A recurring question is whether a centralized or a decentralized approach should be employed for the multi-robot task allocation problem. Respectively, these two approaches improve either solution time or communication cost. The more involved agents and disparate skills the more a decentralized solution is appropriated.

The next subsections address techniques used to plan and negotiate, then, the correlation between the adaptive architecture in Figure 8 and several agent architectures are presented.

3.5.3.1 Planning and Scheduling

When an unexpected event occurs in the environment, the course of actions of an agent may be broken. But, directly reallocate his tasks to another agent is often inadequate because two agents frequently realize the same objective in a different way. Instead, the system has to plan to find a new course of actions that can be performed in this new context. Smith in [23] gives several planning techniques: STRIPS like, Graphplan, SAT, HTN or Markov Decision Process.

As human plays a key role in HoliDes reason about abstract tasks instead of atomic actions may be relevant, therefore, an HTN planner could be a good candidate for tasks reallocation in HoliDes. Moreover, representing existing and valid processes from a constrained domain like aeronautic in the HTN paradigm will be easier than a STRIPS like, SAT, or MDP representation.

Erol in [9] provides the first formal analyze of an HTN planner: UMCP. Otherwise, the HTN planner has to deal with uncertainty, dynamic environment and distributed planning. In this context, Durfee in [6] focuses on HTN planning to perform a continual planning and gives several references : DSIPE and NOAH. We can cite JSHOP and O-Plan too.

Optimize resource and time constraints are frequently viewed as a scheduling problem and solved as a Constraint Satisfaction Problem. In fact, planning involves the use of techniques from scheduling problem, for instance, UMCP uses constraints refinement to cut search space. It may be relevant to split planning and scheduling, like in [26] where scheduling is used by an agent to evaluate new tasks arise from the planning process.

Generally, continuous time and resource constraints are better handled by CSP. Laborie in [14] describes resources availability as a gauge and provides an algorithm to propagate resource constraints in scheduling; IxTeT in [15] is a planner which handles sharable resources and time constraints by interleaving planning and scheduling.

3.5.3.2 Bargaining for task allocation

It is important to underline where negotiation takes place among the techniques employed to solve combinatorial optimization problems. One can distinguish the methods used to solve these problems, as an exact approach (ie: A algorithms) or an approximate approach (ie: genetic algorithms) and how we model the problem, for instance as a CSP or a SAT problem. Negotiation is both a way to represent the problem with utility, costs, preferences and so on, and an algorithm, for instance the Contract Net protocol is a greedy algorithm since at each step we consider the best bid only.

Therefore, negotiation may be viewed as a high level algorithm that allows distribution of combinatorial optimization problems where the objective is to find an agreement between several local optimizations. As stated above, the corresponding MRTA problem of the tasks allocation problem in HoliDes is at least strongly NP-Hard and so, the Contract Net protocol will not be enough to obtain an optimal global solution. But, a solution may be viewed by an agent as optimal and by another as suboptimal, depending of the preferred optimization criterion. Since human is in the loop, it may be more suitable to get a suboptimal solution by negotiation and well understood by all agents than an optimal solution from a centralized process where human operators would face difficulties in knowing which preferences were emitted by which agents.

The contract Net Protocol is the best know market-based algorithm, many extensions are proposed in [3] and a sequence diagram are presented in Figure 11. The monotonic concession protocol and the Zeuthen strategy are presented in [25] and a multilateral version is given in [7].

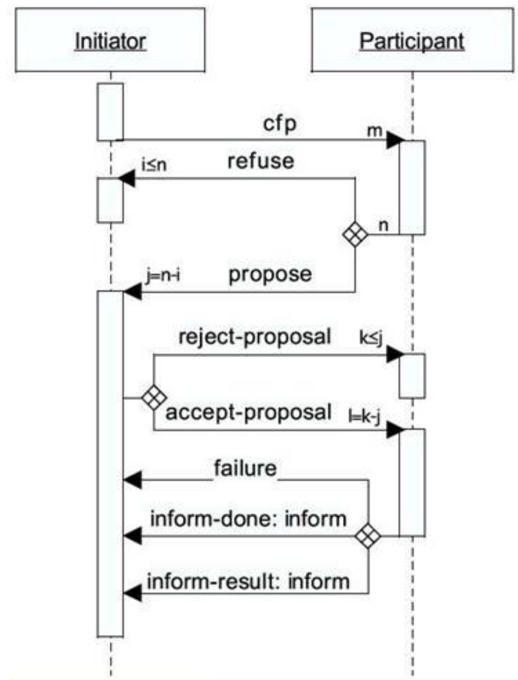

Figure 11: Sequence diagram of the Contract Net Protocol.

3.5.3.3 Architecture

In the simplest case, a task reallocation could be performed as reactive behavior triggered by a new event in the environment. Unfortunately, it will be difficult to see how local changes impact the global system. Therefore, the agent architecture has to deal with reactive and proactive behaviors. In [25], Wooldridge gives two examples of hybrid agent architecture, the Touring Machines architecture and the InteRRaP architecture.

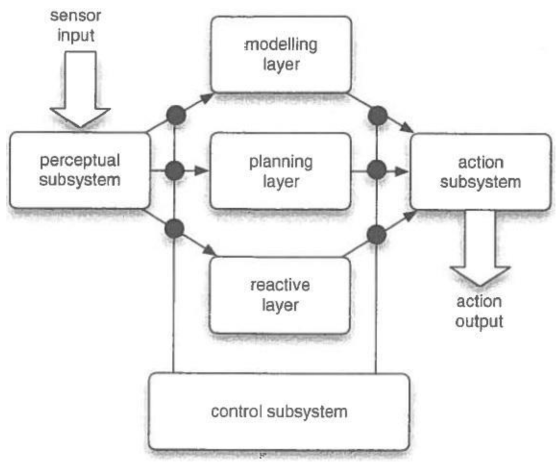

Figure 12 : Touring Machines - an horizontally layered agent architecture

The Touring Machines architecture, showed in Figure 12, is an horizontally layered agent architecture. In this architecture, each layer is connected to the perceptual and the action modules. But, accordingly to Figure 8, the reactive layer should be in charge of the detecting process and the planning layer to the deciding process, however, dispatch data to the reactive or the planning layer involved that the control subsystem handle the detecting process too. And thus, design the control subsystem without duplicate code could be difficult.

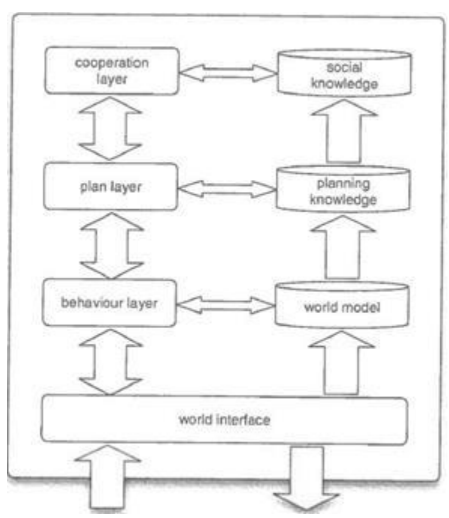

Figure 13: InteRRaP - a vertically layered two-pass agent architecture

The vertically InteRRaP architecture, show in Figure 13, fits better with the adaptation process. Considering the detecting process as a reactive behavior, act directly on the system to adapt in case of emergency is still allowed and proactive behavior can be activated by the reactive layer only.

3.5.4 Landscape of existing software tools

3.5.4.1 Agent-Based Platform

Agent-Builder	- General-purpose - Learning capabilities - Planning capabilities - Multi-Language	agentbuilder.com
Cougaar	- General-purpose - Proven platform - Cognitive Agent - Java	cougaar.org
Jade	- Famous general purpose - Java - Behavioral paradigm - Debug	jade.tilab.com
Soar	- Human performance modelling - Self-healing - Java	sitemaker.edu

3.5.4.2 HTN Planner

UMCP	- Sound and Complete planning - Lisp	cs.umd.edu
JSHOP	- Ordered task decomposition - Symbolic/numeric constraints - Java	cs.umd.edu
O-PLAN	- Distributed planning	aiai.ed.ac.uk

	- <I-N-CA> Constraint Model of Activity - Mixed initiative plan - Lisp	
DSPITE	- Distributed planning - Temporal constraints	[17]

3.5.4.3 Related Projects

I-Globe	- Distributed planning and coordination architecture for dynamic non deterministic multi actor mixed-initiative environment - Flexible planning replanning and task allocation	aiai.ed.ac.uk [13]
I-X: Technology	i-X is a system integration architecture which provides an issue-handling style of architecture, with reasoning and functional capabilities provided by plug in. it allows for sophisticated constraints management and a wide range of communications and capabilities	aiai.ed.ac.uk
K component	Realizing a dynamic software architecture based on adaptation Contract Description Language (ACDL)	[5]

4 Complexity matrix to address adaptation

4.1 Technical

Three main classes of complexity can be considered, from simple to complex. Adaptation that belongs to the first, lowest complexity class should be the easiest to certify. For the most complex cases (e.g., adaptive architecture with learning features), it may be more difficult to certify.

What we are dealing with thus is how the adaptation function produces its output (decision to adapt, "What", "Who", "How") when faced with a change in context. Which mechanisms or techniques can be used to implement that function and what is their respective complexity?

We must be aware that a given AdCos (Adaptive Cooperative System) may implement several of these mechanisms, using for example look-up tables for some aspects, and more sophisticated ones (e.g., rule-based) in the other cases.

4.1.1 Deterministic adaptation

These adaptive mechanisms are based on the following underlying technology:

- *look-up tables*: the output is produced by matching the current context (external and internal) with specific input patterns.
  This is useful for all cases where a specific configuration of the AdCos can safely be associated with specific contexts (external, internal). The AdCos behaves in a fully reactive way. Adaptation of human-machine interaction is frequently based on this scheme (i.e., a given context ⇨ a given display).
- *fully deterministic rule-based systems*: the output is produced by a rule production system that processes both external and internal contexts.

This is useful when more complex contexts must be dealt with and a one to one correspondence (look-up table) cannot be established.

4.1.2 Non deterministic adaptation, but with some levels of predictability and the possibility to certify things

Adaptation is based on the following underlying technology:

- probabilistic or stochastic rule-based systems (e.g., Bayesian networks, Markov models).

4.1.3 Non deterministic adaptation, but with not enough predictability and impossibility to certify

Adaptation involves the following underlying technology:

- neural networks (NN)
- genetic algorithms (GA)
- genetic regulatory networks (GRN)

In many cases, this is expected to be found when the adaptive architecture itself has learning capabilities (i.e., after gaining experience and learning, the architecture will adapt differently when faced to the same inputs. This makes it difficult to predict how it will behave in the future, and therefore difficult to certificate).

4.2 Cognitive

From a cognitive point of view, for the human agents within an adaptive cooperative system, the most important aspect is understanding "Why" adaptation occurs (i.e., why a decision to adapt has been taken) and then for which reasons specific "What", "Who" and "How" have been chosen. It is therefore a question of understanding the full adaptive process.

Clearly, the more complex the adaptation function, from look-up tables to non-deterministic learning systems, the more difficult it is likely understanding adaptation is. Cognitive complexity for human agents is very strongly coupled to their ability to make predictions. Systems whose behavior is hard to predict are cognitively complex.

To make adaptation more understandable to human users (agents), an approach would be to rely on the notion of invariants. The invariants specify some properties of the AdCoS and its relation with its (internal and external) context that must be maintained permanently. For example, *all super-ordinate tasks assigned to the cooperative system must be distributed between the agents*, human and machine, so that at all times the cooperative does what it is supposed to do. The portion in italic is an invariant. If new tasks are assigned to the AdCoS, the invariant will be violated (the new tasks will be unassigned). Adaptation therefore consists in trying to get back to verifying the invariants, in a kind of homeostasis mechanism, where all invariants must be permanently fulfilled (or violated under for the time it takes to get back to homeostasis).

In that case, the "Why" beyond a given adaptation becomes obvious, it's the violation of a specific invariant.

The "What" also is usually clear, because what need to be adapted are the aspects of the cooperative system (in the case of the example, the task distribution) that will bring the system back to compliance with the invariants (homeostasis). The "What" is generally clearly involved in the clause that has been violated.

By relying on such invariants it becomes easier to communicate to the human agents "Why" an adaptation is required (e.g., deviation of the car from its intended lane, violated a lane keeping invariant) and "What" needs to be adapted (e.g., the car's position in its lane). These two aspects can very easily be communicated through appropriate human-machine interfaces (and most car manufacturers do exactly that). Using such invariant is also very useful for verification: such invariant can be translated to a property that must always be true. It can be formally checked through formal proof (like a theorem) or through simulation: at each step, it must be true. If not, a counter example is shown (those that break the property) and so it is very easy to understand the problem (links with WP4).

4.3 Regulatory

As stated above, the three proposed classes of complexity are ranked below in order of decreasing certifiability. As certifiability is strongly linked to predictability, deterministic adaptation is the way to have full predictability, then to be able to certify in all cases.

- *deterministic adaptation*: certifiable
- *non deterministic adaptation, with some predictability*: certifiable in some cases
- *non deterministic adaptation, with no predictability*: generally not certifiable

5 First proposal of specification of « context » functions

5.1 Definitions

AdCos are called to adapt their behaviour according to the circumstances they play in for keeping to achieve their goal in an efficient, effective, and, possibly, improved fashion.

The "**context**" can be defined as the set of factors affecting the AdCos performance and represents the situations the AdCos needs to react to in order to keep in operating in a safe mode (i.e., the "why" of the adaptation in the framework).

Context Assessment is then a necessary stage for triggering adaptation.
It involves the measurement, the interpretation and the prediction of the selected parameters of interest.

The first issue concerns the "Why" of the adaptation and how it could be determined. The Why is based on measuring and/or predicting the internal and the external context as for example the status of an operator or the status of an automated system that can fail or degrade context (e.g. the status of the human operators, or automated systems. Any changes in the internal and external contexts (Why) influence and determine changes in the adaptations (what).

One of the problems regarding context has to do with the exactly definition of its parameters. As a matter of fact, in the literature it is possible to find different definitions, some of them more abstract and some of them more concrete, and that have to do with the technological approach in the representation and elaboration of contextual information. The most used definition in scientific literature describes the context as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."

It can be identified, among the different definitions and interpretations of context, three levels of processing in context awareness: 1) a low level where information are acquired from different sources to extract the features of interest; 2) an intermediate level where a model of context should be identified in order to capture the different information coming from the low level and to integrate them in a coherent framework; 3) an upper level where actions to the context changes are in order to adapt system behaviour.

At the lower level, for example, the most used technologies include signals elaboration, data mining, and classification through machine learning. The intermediate level includes, for example, processing based on rules and ontologies. At the upper level, the adaptation of the behaviour includes procedures developed in a programming language.

One of the most important study concerning context awareness and its component is by [65]. In the study a conceptual model and a software infrastructure are identified. Different components co-occur to the context awareness: widget, which collect information from the environment; interpreters that elaborate them to provide information at an upper level; aggregators that group information regarding similar activities; services to act on the environment and discoverer to find other components/ information present in the environment. However, one of the limits common to these class of studies is that the representation and elaboration of the context is based on ad hoc procedures. Actually, the field of studies is trying to overcome the problem through more general-purpose processing as the ones based on ontologies, an explicit conceptualization, formal and shared, of a knowledge domain.

5.2 External and internal context

In the project, the following elements of context need to be defined and assessed:

- context internal to the agents
    - human agent
        - availability
        - current capabilities
        - current state
            - workload

        - fatigue
        - vigilance
    - machine agent
        - availability
        - current capabilities
        - current state
            - nominal
            - non nominal, with various discrete values (e.g., see flight control laws example in the A320.
- context internal to the cooperative system
    - current task distribution
    - current resource consumption (allocation)
    - current interaction structure (which agents in the system interact together)

This is the context external to the agents and/or cooperative system:

- context external to the cooperative system
    - super-ordinate tasks the cooperative system has to perform
    - resources the cooperative system has to perform the super-ordinate tasks
    - environment in which the cooperative system operates
        - e.g., weather
        - e.g., communication infrastructure (e.g., if it partially fails, the cooperative has to adapt)

The Context Assessment function must therefore be able to assess all these different types of contexts (though in a given type of cooperative system, only a subset of the cases needs to be covered).

5.3 Structure of context functions

The assessment of the context is mandatory for adaptation in cooperative systems. Context must be assessed at all times since changes in context are what drives adaptation. Monitoring the context permanently is therefore mandatory.

Designing the context assessment function means putting in place the sensors and information processing logic needed for that permanent monitoring. The nature of the *sensors* and processing heavily depends on which internal / external context needs to be monitored. For example, when in the Automotive scenarios, several inputs have been identified as context-internal and context-external data. Input data like the facial expression, the gaze scanning, the eye-blink of the human operator under monitoring can be captured by means of *eye trackers*, that should be properly (and possibly seamlessly) placed in the in-vehicle environment. Other bio-metric data can be observed by means of devices like *skin-conductance sensors, EEG, ECG,* and others. Such data can be exploited also to test the effect of the AdCos demonstrator or to prove the validity of the operator models that will be developed in the project. *Microphones* can record the background audio of the environment where the operator acts, by this way providing the sounds that need to be analysed to determine some aspects of the operator' current situation that are considered interesting for adaptation. In simulated environment, *artificial data* can be used in place of real data captured on the field.

The Context Assessment module performs a constant context analysis in real-time, to the aim of providing high level information that are synthetic and meaningful enough to be used within the cognitive control loop.

The raw data captured by sensors can be combined together and processed by using data mining, machine learning and inferences. In these cases, great care should be paid to the computational efficiency of the selected elaboration algorithms in order to allow the context analysis also in presence of time and resource constraints.
The output of the Context Assessment is a Context Annotation, i.e., the meaning of the elaborated information is expressed according to a machine-understandable formalism coherent with an explicit and non-ambiguous semantic that needs to be defined. Examples of output in an internal context-assessment case can be the level of the operator drowsiness or distraction, or some expression characterizing the weather, when in an external context assessment case.

5.4 Methods, Techniques and Tools for Context Assessment

Context Assessment functions are called in the Perception and (partially) in evaluation cognitive stages of the control cognitive loop.
In the following, we present the example of context assessment functions that will be designed and developed within the HoliDes project.

5.4.1 OFF's DIR – Driver's Intention Recognition

In WP3, OFF will develop a context assessment module for the Automotive domain that provides an AdCoS with assessments about the internal context of the AdCoS concerning the human driver's current manoeuvre intentions and driving behaviour (in the following denoted Driver Intention Recognition (DIR) module).

As described in Section 7.4, the overall AdCoS application to be developed for the CRF demonstrator is a unique supporting system that adapts to the behaviour of the different agents, depending on the internal and external conditions. In this AdCoS, the prediction of the driver's intention and behaviour, as provided by the DIR module, will serve as one of several potential “triggers” for the adaptation of the AdCoS application.

The CRF AdCoS consists of four cars with machine agents and human agents inside (Figure 14) travelling on a highway. In the primary use case, car A wants to change the lane to overtake truck C. During this manoeuvre, a collision with the other traffic participants has to be avoided. In the CRF AdCoS, car A will be equipped with several machine agents: a Lane-Change Assistant, an Overtaking-Assistant, and an advanced Forward Collision Warning (FCW) that provides autonomous assisted and emergency braking functionalities. Currently, theses machine agents work without mutual interaction and adaptation. This can lead to unwanted warnings and interventions, which have the potential to annoy the driver to the point of disregarding or disabling the safety device or even introduce new safety critical situations. E.g., as car A approaches the lead-vehicle C to start the overtaking manoeuvre, its driver can get a warning and possible intervention from the FCW due to the decreasing distance to C.

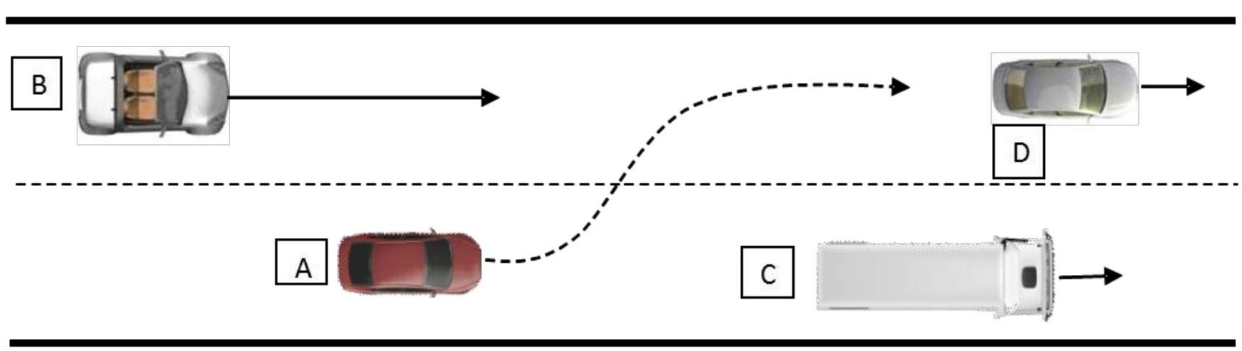

Figure 14 : Representation of a use-case for the CRF AdCoS.

As a solution, the machine agents on board of car A should have an assessment of the unobservable intentions of the driver. To achieve this, OFF will extend the CRF AdCoS application by a DIR module that provides the

AdCoS with predictions about the driver's manoeuvre intentions and future driving behaviour.

In this context, manoeuvre intentions are defined as the unobservable intentions of a human driver to currently or in the near future perform one of a set of high-level manoeuvres, denoted by *B*, as defined in a skill hierarchy. Figure 15 shows an exemplary skill-hierarchy applicable for the WP9 use-case, where the complex driving behaviour for driving on highways can be represented by the four manoeuvres for performing lane changes to the left lane, lane changes to the right lane, lane-following, and car-following.

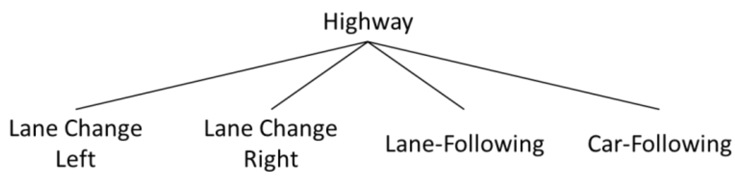

Figure 15: Exemplary basic skill-hierarchy.

If necessary, each of these manoeuvres could be further decomposed into simpler low-level manoeuvres, such as emergency brakes and distance-keeping in the case of car-following, or decomposing a lane-change into the phases preparation, the lane change itself, and the final realigning of the vehicle in the new lane Figure 16.

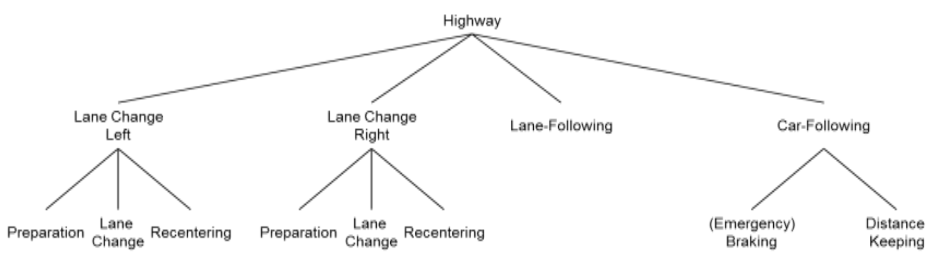

Figure 16: Exemplary advanced skill-hierarchy.

Driving behaviour is defined as a sequence of actuator actions, denoted by *A* (we assume steering wheel angles for lateral control and acceleration-/braking-pedal positions for longitudinal control) that is expected to be observed during a specific manoeuvre in the skill hierarchy.

Figure 17 shows the expected architecture of the DIR module. The module will primarily consist of three components: A probabilistic model of the human driver, based on and extending previously developed Bayesian Autonomous Driver Mixture-of-Behavior (BAD MoB) models, an inference engine, and an adaption manager.

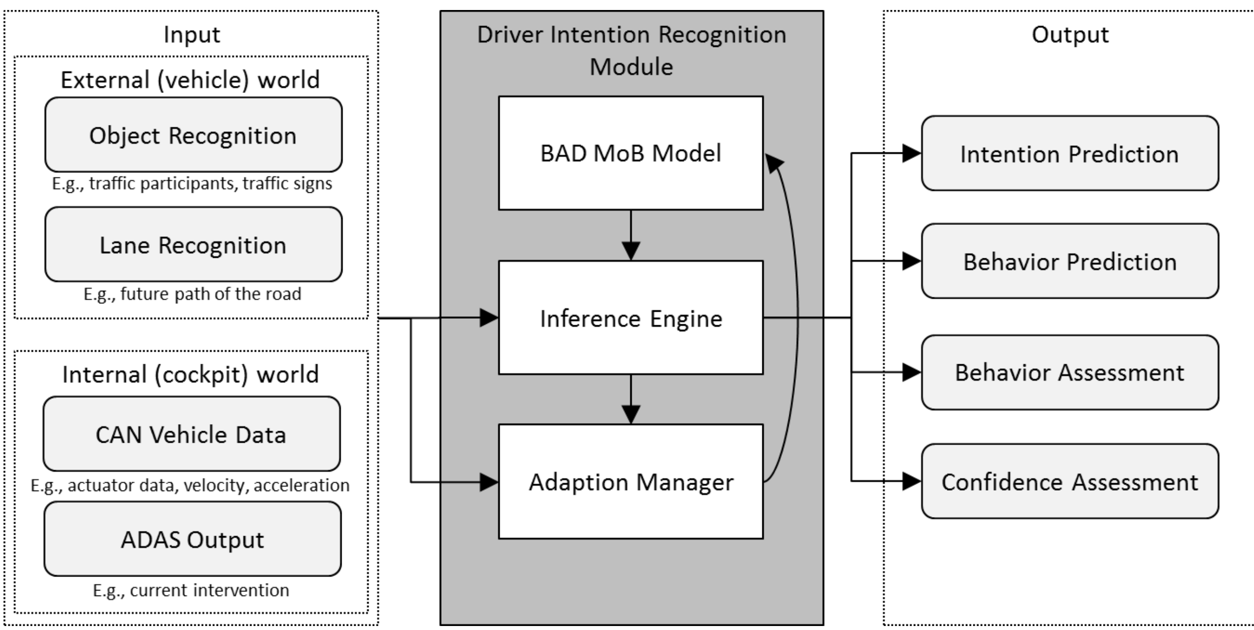

Figure 17 : Overview of the architecture of the DIR module.

BAD MoB Model:

BAD MoB models are probabilistic model of the human driver based on (Dynamic) Bayesian Networks. They will primarily be developed in WP2, while their utilization in the DIR module will be investigated in WP3.

A Bayesian Network (BN) is an annotated directed acyclic graph (DAG) that encodes a joint probability over a set of random variables $X = \{X_1, \ldots, X_n\}$. Formally, a BN B is defined as a pair $B = \{G, \theta\}$.The component G is a DAG, whose vertices correspond to the random variables $X_1, \ldots, X_n$, and whose arcs define the (in)dependencies between these variables, in that each variable X_i is independent of its non-descendants given its (possible empts) set of parents $Pa(X_i)$ in G. The component θ represents a set of parameters that quantify the probabilities of the BN. Given G and θ, a BN B defines a unique joint probability distribution (JPD) over X as:

$$P(X) = \prod_{i=1}^{n} P\big(X_i | Pa(X_i)\big).$$

DBNs extend BNs to model the stochastic evolution over a set of variables $X = \{X_1, \ldots, X_n\}$ over time. A DBN D is defined as a pair $D = \{B^1, B^{\rightarrow}\}$, where $B^1 = \{G^1, \theta^1\}$ is a BN that defines the probability distribution $P(X^1)$ and, under the assumption of first-order Markov and stationary processes, $B^{\rightarrow} = \{G^{\rightarrow}, \theta^{\rightarrow}\}$ is a two-slice Bayesian network (2TBN) that defines the conditional probability distribution (CPD) $P(X^t | X^{t-1})$ for all t. The nodes in the first slice of the 2TBN do not have any parameters associated with them, but each node in the second slice of the 2TBN has an associated CPD which defines $P\big(X_i^t | Pa(X_i^t)\big)$, where a parent $X_j^t \in Pa(X_i^t)$ can either be in time-slice t or $t-1$. The JPD over any number of T time-slices is then given by:

$$P(X^{1:T}) = \prod_{t=1}^{T} \prod_{i=1}^{n} P\big(X_i^t | Pa(X_i^t)\big).$$

A BAD MoB model is a DBN that implements the complex sensorimotor system of human drivers in a modular and hierarchical probabilistic architecture by combining multiple nested DBNs with distinct purposes. A BAD MoB model is based on the assumption that the complex driving competence of a human driver can be described by a skill hierarchy that that hierarchically decomposes complex high-level driving behaviour or manoeuvres into simpler, or pure driving behaviours (c.f., Figure 38). Each basic skill in the skill hierarchy is realized by a distinct *action*-model that implements the isolated sensorimotor schema of the corresponding driving skill, i.e., the relation between driving actions A and the available observations from the environment, denoted by O . The appropriateness of a pure or a mixture of basic skills in a given situation is inferred by a *behavior-classification*-model. The functional interaction of action- and behavior-classification-models then allows the context-dependent generation, prediction, and assessment of complex human driving behaviour and intentions.

The primary functionality of a behaviour and intention prediction is to provide an estimate of the operator's current intention in respect to a preliminary defined set of potential intentions.

Inference Engine:

During runtime, the inference engine will utilize the BAD MoB model to answer probability queries about the desired output using the actual input as evidence.

Adaption Manager:

Within the CFR AdCoS, the DIR module can be seen as a further machine agent. As such, it is subject to the adaption of task performance as described in Section 2.1.2.2. During runtime, the adaption manager will continuously assess the input and output of the AdCoS component to recalibrate the parameters of the model during runtime, in order to adapt the model to the actual driver and, over time, achieve a better performance. For this, new techniques will be developed by OFF in WP3 to assess the current performance of the intention prediction.

Input:

On a generic level, a context assessment module for intention prediction module should utilize input in the form of information about the operator, the operated system, and the environment. In the case of the DIR module, this input will be provided by external sensors (e.g., cameras and radar) that perceive the environment and internal sensors (e.g., controller area network bus and cameras) that observe the driver and the vehicle. The expected necessary input includes but may not be limited to:

- Information about recognized objects like e.g., surrounding traffic participants and traffic signs,
- Information about the future path of the road, including e.g., the distance from lane edges,
- Information about the current state of the car, like e.g., current velocities and accelerations,
- Information about the current state of the actuators, like e.g., steering wheel angles and pedal positions,
- Information about the current state and outputs of other machine agents.

Output:

The primary output of the machine agent consists of sets of temporally evolving belief state estimates in the form of CPDs of the driving behaviour, resp. the driving actions, *A* and manoeuvre intentions *B*, given the all current sensor observations *O*. In the following, the planned outputs, the machine agent will provide to the overall AdCoS shall be briefly introduced:

Maneuver Intention Classification / Prediction:

At each time-step t, the context assessment module will provide a maneuver intention prediction via the CPD $P(B^{t+n}|a^{1:t}, o^{1:t})$, where n is a desired anticipatory horizon. If $n = 0$, this can be seen as a classification of the current maneuver intention.

Lateral and Longitudinal Driving Action Prediction:
At each time-step t, the context assessment module will provide an action prediction via the CPD $P(A^{t+1+n}|a^{1:t}, o^{1:t})$, where n is a desired anticipatory horizon.

Likelihood of the current driving actions:
At each time-step t, the context assessment module will provide the log-likelihood of the last m chosen driving actions $\log P\left(a^{t-m:t}|a^{1:t-(m-1)}, o^{1:t}\right)$. Under the assumption that the model represents normative driving, this can be used as a measure of normative driving. Low values or sudden drops (under the assumption that certain thresholds are defined) indicate that the driver does not show normative driving behavior and can be used as a further trigger for needed adaptation of the AdCoS.

Confidence:
At each time step t, for each provided output, the context assessment module will provide an assessment of its confidence in the inferred outputs. Note that this confidence does not relate to the probabilities itself (which are obvious from the outputs) but takes into account the confidence in the estimated parameters used for the inference. Higher confidence values indicate the confidence of the context assessment module in the correctness of the inference and should rise in the presence of more available data.

5.4.1.1 Development workflow

Figure 18 shows the tool-chain expected to be used for the development of the DIR module and its connection to the CRF AdCoS tool-chain. Primarily, the development of the DIR module will be linked to the CRF tool-chain via RTMaps.

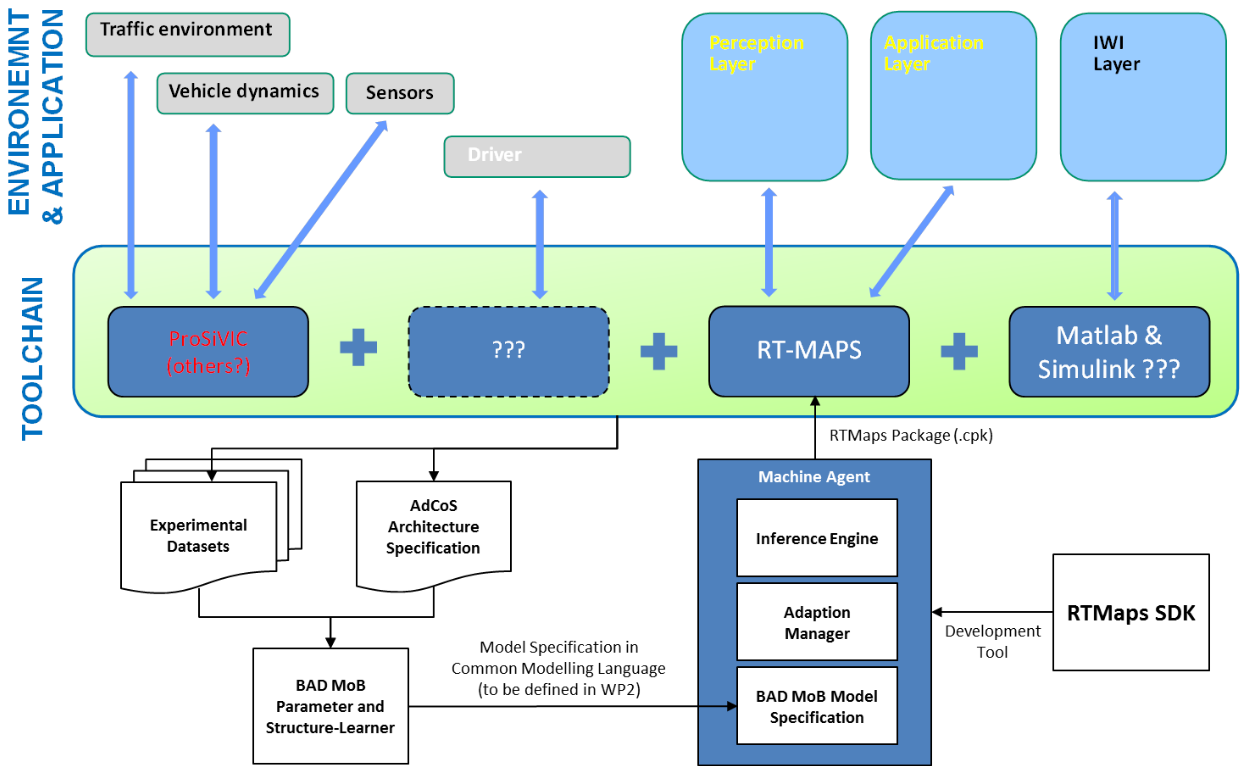

Figure 18 : CRF tool-chain for design, development and testing of AdCoS including the DIR machine agent (before going on real vehicle).

At each stage during the development of the CRF AdCoS, the current specification of the AdCoS architecture defining e.g., available inputs of the perception layer, will be used to derive the potential graph-structures for the BAD MoB model.

The actual graph structure and parameters of the BAD MoB model are learned via machine-learning methods. The algorithms and learning procedures will be implemented in proprietary software developed by OFF (BAD MoB Parameter and Structure-Learner). The output is a fully defined specification of a BAD MoB model, essentially consisting of a description of the graph-structure and the parameters of the model. The specification will be described using the yet to be defined common modelling language developed in WP2. For learning the structure and parameters of an initial BAD MoB model, experimental datasets of time-series of data samples in the same format as expected during runtime is required. These could be obtained in simulator studies using driving simulators available in HoliDes utilizing ProSIVIC for simulating sensors, or in real-life driving studies using the CRF demonstrator vehicle.

The actual DIR machine agent will be developed in WP9 using the RTMaps SDK for Visual Studio. The output is a RTMaps component package (.pck) which can then be used within RTMaps for the overall development and simulation of the CRF AdCoS.

5.4.1.2 Covered Requirements

Table 2 shows the list of requirements addressed by the development of the DIR module in WP3.

REQ-ID	Name
WP9_CRF_AUT_REQ3_v1.0	Classification of driver's cognitive state
WP9_OFF_AUT_REQ1_v1.0	Offline parameter and structure learning
WP9_OFF_AUT_REQ2_v1.0	Online parameter learning and adaptation
WP9_OFF_AUT_REQ4_v1.0	Guaranteed maximal computation time
WP9_OFF_AUT_REQ7_v1.0	Manoeuvre Classification
WP9_OFF_AUT_REQ8_v1.0	Manoeuvre Intention Classification
WP9_OFF_AUT_REQ9_v1.0	Driving Style Classification
WP9_OFF_AUT_REQ10_v1.0	Likelihood of current driving behaviour
WP9_OFF_AUT_REQ11_v1.0	Confidence in Manoeuvre Classification
WP9_OFF_AUT_REQ12_v1.0	Confidence in Intention Classification
WP9_OFF_AUT_REQ13_v1.0	Confidence in Driving Style Classification

Table 2: Requirements addressed by the development of the DIR module in WP3.

Furthermore, the methods, techniques and tools related to the DIR module could be helpful in fulfilling the requirements shown in Table 3.

REQ-ID	Name
WP6_AWI_HEA_REQ1_v0.2	Identify the operator's experience level
WP7_HON_AER_REQ78_v0.1	Evaluation of agent action
WP7_HON_AER_REQ87_v0.1	Classification of physiological output
WP8_ADS_CTR_REQ15_v0.1	Operator state assessment
WP9_TWT_AUT_REQ04_v0.1	Distraction level classifier algorithm for feedback app
WP9_DLR_AUT_REQ01_v1.0	Learning of individual driving behaviour
WP9_DLR_AUT_REQ02_v1.0	Online learning
WP9_DLR_AUT_REQ03_v1.0	Offering save maneuvers

Table 3 : Requirements addressed by the development of BAD MoB models and the machine agent for intention and behaviour prediction.

5.4.2 IFS DMF – Driver's Monitoring Function

The IFS Driver Monitoring Function (DMF) will be in charge to supervise the driver (more precisely, the driver simulated with COSMODRIVE) in order to jointly assess (1) human errors, like an inadequate visual scanning, an erroneous situation awareness or a risky driving behavior, and (2) the situational risk.

Such information will be exploited to adapt the AdCoS and to support the driver (i.e., COSMODRIVE) to avoid accident due to the assessed human errors (misperception, situation misunderstanding, risky behaviour, etc.).

The DMF will be based on a State-Transition Graph model in charge to provide in real time some assessment values of drivers' errors and risk, together with his needs and difficulties.

Then, these "assessment values" will be used in the AdCoS for (1) activating technical aid or not, and (2) determine the best way to interact with the COSMODRIVE driver (i.e., adapt HMI modalities according to the context, ranging from car control taking to information delivery or warning signals).

The DMF core scenario is focused on the lane change maneuver. In this specific context, DMF will have (1) to observe and monitor COSMODRIVE driving behaviors and (2) to diagnose risky behavior maneuver and/or to assess situational risk (e.g. intention to implement the lane change in a critical time) and (3) to adapt the AdCoS (virtually simulated on RT-MAPS) in the right way to adequately support the COSMODRIVE driver and avoid accident.

5.4.3 TWT Driver’s distraction estimator

TWT will study how to estimate the driver’s distraction on the basis of auditory and visual information captured within the vehicle.

Auditory information captured by means of microphones are going to be analysed in order to determine the nature of the source, being it the conversation with other passengers, or playing music, or background noise coming from an open window.

Also the tone of the conversation will be taken into consideration when observing the effects of such auditory stimuli on the driver's distraction.

As to the visual information, facial expression, head posture and eye gaze captured by means of in-vehicle cameras will be considered as the input of the estimator as well.

6 First proposal of specification of « adaptation computation » functions

Building adaptive systems involves two main challenges: (1) inferring the current state of the overall system, its components and its environment and (2) using this information to derive re-configurations in order to "optimize" the overall system according to a trade-off of different relevant parameters.
Nowadays, autonomous systems increase constantly and 'step by step' become part of our everyday life. The "ambient intelligence" vision described by Weiser [44] depicts a not so future society where people and objects (household appliances (Forlizzi [39]), cars, clothes, public transports, computers…) are able to interact with each other's and also with their environment.
But the thing is that today, our knowledge in technology and algorithms doesn't allow us to develop fully reliable autonomous systems to figured out all possible situations (Zieba[46]). We could compromise by developing adaptable interactions aiming to "optimize" the use of competencies of humans and machines. Such adaptable interaction abilities would allow a collaborative control that will show real time configuration of autonomy (and interactions) in order to maintain acceptable level of efficiency in every kind of strength (Zieba[46]).

6.1 Context for adaptation

The first (1) key issue to an achievement consist on be able to approach as a closest possible state assessment of the current situation. The "current situation" defined in Holides for the AdCoS concept regroups two scientific domains known as "situation awareness" (Endsley [8] & [37]) and "context-aware" (Dey [36]).

Situation awareness (SA) is often quoted as one of the most important indicator for human factor activities but SA is difficult to measure and qualify. SA by Mica Endsley ([8] & [37]) is a three parts concept which includes the levels perception, comprehension and projection. She defines SA as "the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future" as the most encountered understanding of SA in industry.

See section 5 for more details on a first proposal of specification of "context" functions

6.2 Adaptation functions

The second (2) key issues to achieve the adaptation are to be able to compute the adequate reconfiguration to fulfil the global mission of the human and machine agents. Hollnagel [41] has introduced the pro-active organization notion (Figure 19).

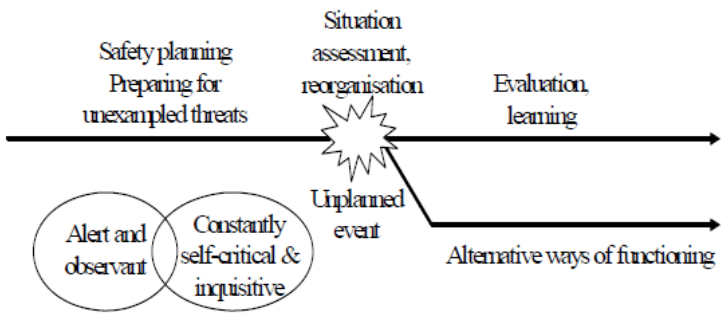

Figure 19: Pro-active organisation from [Hollnagel, 2006]

Pro-active organization and system adaptation capabilities are related concept named "resilience". Wreshall [45] defines the resilience as "the intrinsic ability of an organization to keep or recover a stable state allowing it to continue operations after a major mishap or in presence of a continuous stress". Fiksel [38] proposes four characteristics of resilience: diversity, efficiency, adaptability (capacity of the system to adapt with new events), cohesion. The table below shows the different characteristics of a resilient human-machine system:

	Diversity	Efficiency	Adaptability	Cohesion
Human	Different strategies	Efficient decisions	Human adaptability	Respect of objectives
Machine	Material and decisional redundancy	Efficient decisions	Adjustable autonomy	Respect of prescribed plan of action
Human-Machine system	Different mode of human-machine cooperation	Efficient cooperation	Adjustable autonomy	Efficient communication

Table 4: Characteristics of resilience for human-machine systems adapted from [Fiksel, 2003] and [Zieba, 2009]

Collaborative control could be defined according to the Scerbo [43] taxonomy that describes adaptable and adaptive systems: in adaptable system, the allocation functions of the systems are triggered by the human operator. In adaptive system, both human and system have the same interaction level. In reality, the collaborative control should manage transitions between different ways of autonomy. The task of the human (or machine) is more or less critical according to the control level: (i) strategic level: global objectives, (ii) tactical level: sub goal elaborate in the strategic level, (iii) operational level: how are realized the tactical objectives. Human and machine interact in different mode. These interactions have been defined by Parasuranam [42] in four different categories: (1): information task, (2): information analysis, (3): decision-making, (4): action implementation. From that assumption, Zieba [47] proposed a framework for resilient human machine cooperation to adjust the level of autonomy according to the context.

In several WP6-9 use cases as in most Human – Machine interactions, there is a high level of complexity due to the heterogeneity and the diversity of actors and data, due to the dynamicity of the application environments and due to the non-deterministic human behaviors. Thus, the scientific community has been interested, for the last several years, in the development of new solutions based on computation distribution and control decentralization, which are more appropriate for solving such problems (Silaghi [48] & Valchenaers [49]).

Self-organizing multi-agent approach is based on the emergence of a functional structure spontaneously maintained in a dynamic equilibrium by all participating components (Heylighen [50]). As described in Serugendo [51], self-organizing MAS (Multi-Agent Systems) offer opportunities to simulate real complex systems, because agents have ideally autonomous behaviors, adapt constantly their state relatively to each other, learn from experience, and dynamically create group and organization.

The choice of adaptive multi-agent systems technology is relevant, because of its ability: to represent the knowledge of actors/components locally (constraints and objectives); to be distributed (system components are distributed); to be evolutionary (system components have their own life cycle and decision process); and to be open (system components can appear and disappear).

The approach that we will describe comes from AMAS (Adaptive Multi Agents Systems) (Glize [46]) and based on the theorem of Functional Adequacy. The functional adequacy could be summarizing in a high level description as “having the appropriate behavior for a given task”.

Theorem: For any functionally adequate system, there exists at least one cooperative internal medium system that fulfills an equivalent function in the same environment.

Definition: A cooperative internal medium system is a system where no Non Cooperative Situation (NCS) exists.
We will detail the NCS in the cooperation module.
Therefore an adaptive collaborative system functionally adequate is a system that may satisfy the four following properties:

- **Sincerity**: if a proposition / information is true, an agent couldn't say anything different
- **Willingness**: No resulting prejudice for an agent's action
- **Fairness**: We always try to satisfy the agent with the higher criticality
- **Reciprocity**: Each agent knows that it as well as others agents verify these three main properties.

Based on that theorem, we will describe the architecture of an AMAS agent (Bernon [43]).
First, each agent has the four following characteristics:

- An agent is **autonomous**, it could make decision on its own
- An agent **isn't aware of the global function** of the system (the global function is an epiphenomena of the overall collaborative and adaptive multi agent system)
- An agent is able to **detect non cooperative situations** (see cooperative module)
- An agent is **benevolent**: it tries to achieve its own objective while helping others agents (agent in a more critical state)

Second, each agent has several modules as show in Figure 20

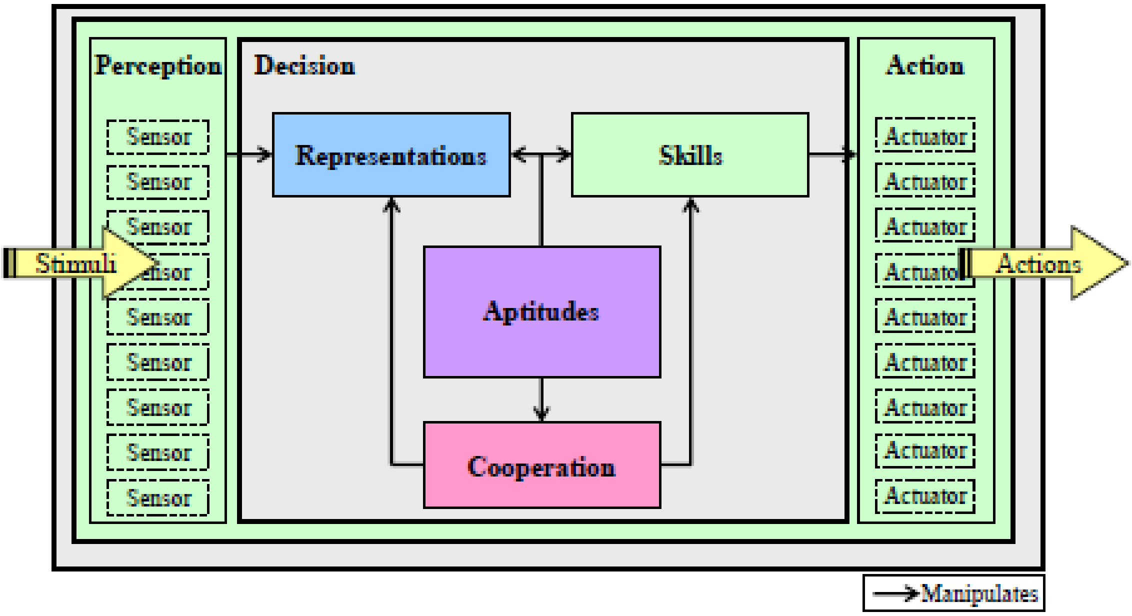

Figure 20: A cooperative agent [Bernon, 2004]

The perception module: the input received by the agent. These inputs could have different natures: Boolean, real value, complex messages…

The Action module: the output of the agent. Outputs could as well have different natures: Boolean, real value, complex messages…

The Representations module: is the belief of the agent has on the environment. It includes also the representation of the others agents as well as the representation it has on itself: internal (itself), external (the environment) and social (the others agents),

The skills module: is the knowledge of the agent that could contain its own objectives. Skills can be classical rule based or more advanced cognitive network.

The aptitude module: is the set of methods and tools to accomplish the agent's treatments.

The cooperative module: is the module that manipulates Representation and Skills to detect, repair and anticipate Non Collaborative Situations (NCS). NCS could have different forms:

- **Incomprehension** and **ambiguities** that come from the perception module in which an input message could be not understand clearly
- **Incompetence** and **unproductivity** that come from the decision module in which the agent couldn't treat the information or couldn't produce useful output.
- **Uselessness**, **conflict** and **concurrence** that come from the action module in which the agent considered itself useless or if the action provides by the agent is in conflict with another one or if the same action could be provided by another agent in the same time.

7 Some examples of adaptation for each domains

7.1 Health

7.1.1 Introduction

The Guided Patient Positioning System is the AdCoS that provides guidance to the operators during preparing and positioning patient for MRI examinations.

Figure 21: Example illustrations of an MRI

Correct positioning of the patient for the MRI examination and using the right coils and other devices are important to get good diagnostic quality images, but also important to avoid safety issues. Currently the operator is trained for this. The on-line guidance system intends to improve usability and to reduce risks, also in case of novice, less experienced users.

7.1.2 Available information

The Guided Patient Positioning System has access to the following on-line information, which can be used in the context assessment part of the system:

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

7.1.3 Adaptivity

The Guided Patient Positioning System shall provide on-line guidance and actual information during positioning of the patient.

It needs to use the input data, as listed above. From the patient characteristics and MRI examination procedure the system can derive the instructions to the operator, which needs to be updated on-line based on detectable actions by 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.

7.2 Aeronautic – Diversion assistant (DivA)

7.2.1 Description

DivA monitors the current aircraft state (e.g. aircraft position, performance, flight plan, etc.) and by comparing it with relevant pieces of static information (e.g. navigation database, charts, etc.) and dynamic information (e.g. strategic weather, DNOTAM, etc.), it calculates paths to available diversion airports. Considering the state of the pilot (being also monitored) and dynamic changes in the environment and aircraft, DivA defines priorities for each potential diversion airport and presents the prioritized list of potential diversion airports to the flight crew. The list is supplied with other relevant information in predefined selected categories (e.g. distance of the airport from the current position, weather at the airport, runway length, approach type, etc.).

7.2.2 Why to adapt?

For DivA there are the following main situations that should trigger the adaptation of the system:

1. A change in operator state is detected. This can be a change in workload, physical capacity, fatigue or attention. The change can be in both directions – increasing or decreasing the property.
2. The operator can commit an error during his working procedure.
3. The aircraft can enter a non-normal state due to system failure.

4. The environment can change in a way that affects the ability of the operator or the aircraft. In aeronautics it can have natural cause – weather, ambient light, contrast, visibility or vibrations/turbulences, or manmade cause – traffic information or availability of various services (such as various airport/airways parameters).

7.2.3 What and how to adapt?

DivA should be implemented as EFB solution. EFBs are strictly regulated with respect to saliency and warnings. They should not mirror critical information from the avionics. The remaining means of adaptation are

- prioritization of information to be shown (highlights, list reordering)
- amount of information to be shown
- displaying cues to aspects that have changed
- information sharing/splitting between the pilots.

7.2.4 Current Solutions

Current situation with respect to triggers is
Ad 2.1 and 2.2: operator state or procedure performance is not monitored at all, except for crew cross-checks.

Ad 2.3: system failures are communicated through controls in the flight deck. Crew alerting system produces messages about aircraft system failures. The amount of messages can be high and some attempts to introduce prioritization have been made.

Ad 2.4: some properties are monitored – high contrast edges on displays, day/night operations etc. Others are not or the support is being developed.

7.2.5 Proposed AdCoS Solution for HoliDes

For DivA the following adaptation solutions should be developed and tested:
Ad 2.1: HoliDes methods for operator state inference should be evaluated (physiological metrics, camera monitoring, voice detection). The final

demonstrator solution should minimize intrusiveness of the monitoring so that it can be applied in long cruise phases of the flight.

Ad 2.2: to be further decided – reliability of methods applied in flight is questionable, development needs to be observed.

Ad 2.3: DivA is supposed to give details on which it bases its prioritization. The amount of information will be optimized by situation (environment, aircraft and operator).The prioritization itself contains aspects of adaptation.

Ad 2.4: system based on video recording and recognition is planned to identify suboptimal display layout due to change in lightning, vibrations etc. Automatic adjustments linked to pilot acceptance will create a learning system that should be able to react appropriately (acceptably for pilots) to the situation.

7.3 Control rooms

Concerning control rooms domain in border security operation, some adaptation mechanisms could be envisaged for the following requirements:

- Operator Physical and Mental State Assessment
- Load Balancing on Operator Level
- Assisted User Categorisation
- Layered Help Functions
- Adapting the System to Local Requirements

More details extracted from CAS internal document are presented below:

Operator Physical and Mental State Assessment

Why to adapt? : An adaptive system can increase the effectiveness of the border security operation if operators are in a state that allows them to effectively and efficiency respond to events. If operators are in a sub-optimum state (e.g. absent from workplace, tired or asleep), the supervisors can take measures to guarantee the effectiveness of the control station.

What to adapt? : 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.

The system can respond by taking action, including measures to motivate the operator to remedy the situation (e.g. by using remote actuators) or by notifying the supervisor.

How to adapt? : The system monitors the operator's behaviour with sensors (body-shape recognition and eye tracking) in a non-obtrusive way and that does not violate local ethical or data protection standards and that maintains the individual's dignity.

Load Balancing on Operator Level

Why to adapt? : The system can increase the effectiveness of the individual operators if the workload for each operator is kept in an optimum range.

What to adapt? : The system is able to recognize the load of a single operator compared to the overall loads of all operators in one headquarter.

- The system shall be able to rebalance workload among all available operators when one of them runs in an unforeseen overload scenario (e.g. a low priority event escalates and consumes all the time and attention of its operator).
- The re-assignment can be performed either with or without involvement of the supervisor.

How to adapt?: The system is be able to detect the workload of each individual operator.

- The objective workload is operationalised as the number and criticality of events the operator has to handle;
- The subjective workload is operationalised as the objective workload taking into account additional mediating subjective factors such as current level of fatigue, current stress level and level experience with the current position.

Assisted User Categorisation

Why to adapt? : The system currently does not differentiate between operators with different degrees of experience. A mechanism for supporting the centre management in assigning levels of expertise to individual operators can help increasing the operators' effectiveness and thereby the effectiveness of the entire border security operation. The levels of expertise are:

- 'basic experience': These are entry-level operators with little training and little first-hand experience on the job and who still require intense supervision.
- 'advanced experience': These are operators with solid training and experience who can be trusted to perform reliably in everyday situations.
- 'expert experience': These are operators with the highest degree of expertise who can be relied upon to correctly assess and respond to critical and/or unusual situations and circumstances and who are qualified for the supervision of operators with 'basic experience'.

What to adapt? : The system uses a number of parameters to propose to the centre management the initial level-of-expertise category or a category change for each member of staff. The level-of-expertise categories can be used to offer a layered help function, propose a training measure (recurring training or next-level training) or a personal fresh-up training session with a supervisor, and to be used in the assessment of the subjective workload.

How to adapt? : The system uses the various parameters (and others to be specified) to propose to the centre management the initial category or a category change for each member of staff. Those variables include the time spent in current position, training levels achieved, the number of regular and critical instances mastered the number of faulty decision over a defined period of time, and the performance assessment by supervisor.

Layered Help Functions

Why to adapt? : The system currently offers limited help functionality (mainly access to "help manuals" and bubble help on mouse rollover). Help information should be provided in an easier and more accessible way thereby increasing the operator's efficiency and effectiveness.
What to adapt? : In order to provide access to help information more easily and thereby increasing the operator's efficiency and effectiveness, help should be provided in a form that is based on the industry standard and tailored to the operator`s level of expertise.

How to adapt? : When a user logs into the system, his current level of experience as stored in his user data sets the help functionality to one of three levels of detail:

- Basic-level help: help is given in a way that is easy to understand and covers the basic functional concepts
- Experienced-level help: help covers basic functional concepts as well other, less frequently used or more complex concepts

- Expert-level help: help covers the entire functionality with every option and detail

In all instances, 'more details' can be requested for accessing further information about the current help topic.

Adapting the System to Local Requirements:

Why to adapt? : The system currently offers limited localization features. Any customization has to be hard-wired into the design of workplaces and PC user interfaces. This is costly and adds complexity to the system design.

What to adapt? : The system has pre-defined variants for aspects of the HMI that are related to the local culture, data standards, and regulations in the customer's country. The process of the customization of localization features includes three steps:

- The analysis of relevant customization features (e.g. language, colours, data standards for numerals, time, currencies, etc.) in a given application context (e.g. border security)
- The collection (e.g. in templates) of instantiations of customization features for a number of representative or likely customer cultures / countries
- The identification of features in a given application (e.g. border security application) that are candidates for localization.

Areas for the adaptation of localization features include:

- Interface language: the system supports the language(s) agreed between customer and technology provider;
- Data standards: the system supports the locally applicable standards for numerals, time, calendar dates, keyboards, currencies, weights, lengths, and other measures;

- Use of colours, symbols, concepts and terminology: the system supports options that are expected resp. understood in the customer's culture.

How to adapt?

The adaptation of localization features can occur at compilation time and at run time:

- The application product allows the generation of a customer-specific version of the product that instantiates the applicable customization features (localization generated at compilation time).
- A set of localization features can be invoked by the user as a SW option or as a data format variant (localization generated at run time).
- A set of localization features can be selected at the level of the GUI allowing the parameters to be displayed in different data standards (e.g. selection of nautical miles or kilometres).
    - 

7.4 Automotive

The AdCoS implemented in CRF test-vehicle (TV) is a unique supporting system, adapting to the behaviour of the different agents, depending on the internal and external scenarios. In particular, the following functionalities are implemented:

- Lane-Change Assistant (LCA).
- Overtaking Assistant (OA) – which is a kind of extension of the previous one.
- Extended Forward Collision Warning (eFCW), including assisted braking and, optionally, automatic emergency braking.

In particular, the scenario is represented in Figure 22, consisting of four cars with machine agents (e.g. PADAS) and human agents (drivers):

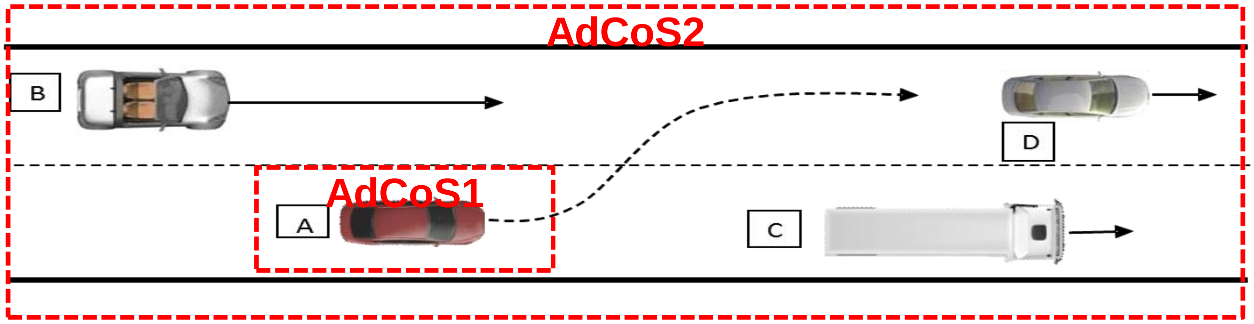

Figure 22: Visual representation of the AdCoS scenario for CRF test-vehicle.

The adaptation is carried out at a twofold level, being based on the external situation (e.g. a vehicle approaching from the rear the ego-vehicle when its intention is to overtake) and, above all on the internal situation (e.g. the driver is distracted from the on-board infotainment system).

In order to accomplish this idea, we have adopted a statistical approach for the co-pilot, which constitutes the core of the AdCoS: the principle is to model our system as a Markovian Decision Process (MDP), to construct optimal warning and intervention strategies (WISs).

In this context, the classification of driver's cognitive state is the "trigger" for the adaptation. In fact, depending on the fact that the driver is distracted or not, the strategies of the AdCoS are modified, both for LCA and for FCW functions. The following schema sketches how this works:

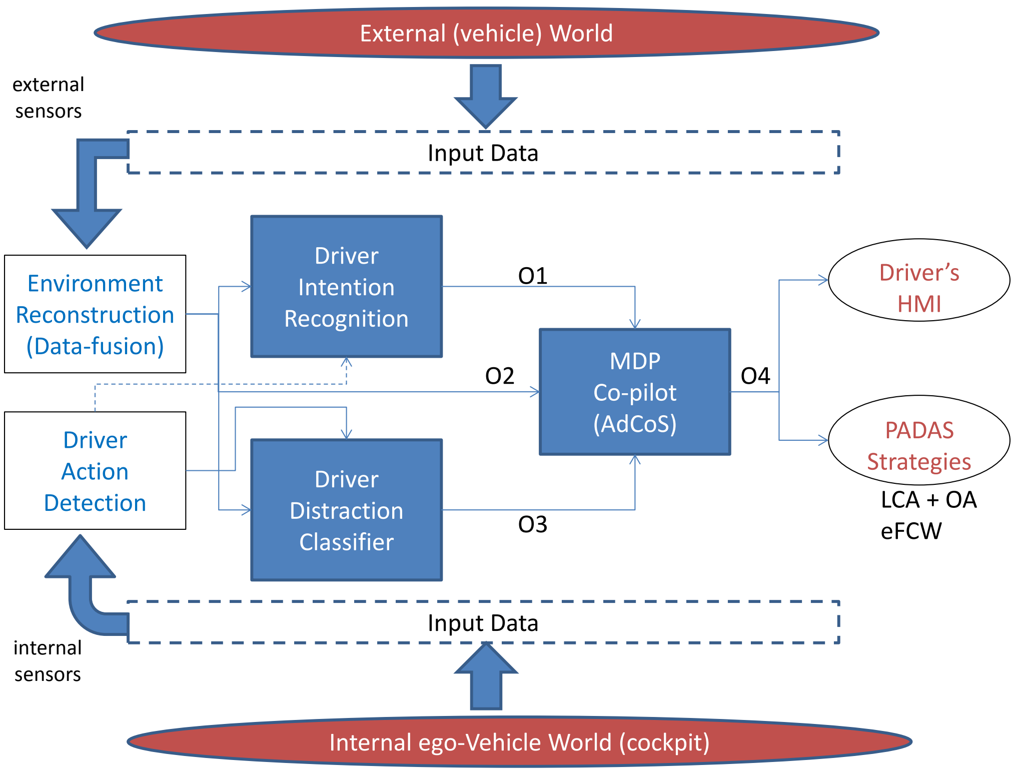

Figure 23 : Sketch of the AdCoS functionality.

With reference to Figure 23, the real world is detected by the on-board sensors (e.g. steering angle, yaw-rate, etc.) and ADAS sensors (e.g. Radar, Camera, etc.), while the detection of the internal world (internal to the test-vehicle, namely the cockpit) concerns the actions of the driver, where he/she is looking at, and so on (pedals position, eye-tracker, etc.). The data from the real-world are put together by the data-fusion (DF) module, which provides the list of obstacles (with a selection of the most relevant ones), the road curvature ahead, the presence of the lanes, and so forth (outputs O2).

In addition, all these data are then used by the **Driver Intention Recognition** (DIR) module and by the **Driver Distraction Classification** (DDC) module, as illustrated in the figure (O1 and O3 outputs, respectively).

The DIR aims at predicting the driver intention that is the manoeuvre which the ego-vehicle – and thus its driver – intends to do. The DDC module provides information about the driver cognitive state, in particular if he/she is distracted or not (at the moment, it is not defined yet if the classification is on a binary level or on more levels).

O1, O2 and O3 are the inputs to the **Co-pilot** module, whose main output (O4) is represented by the computation of an "optimal" manoeuvre, based on the external situation and on the driver state. This manoeuvre is suggested from machine-agent to human-agent, by means of specific warnings, advice and information, according to the cognitive state and intentions of driver, as well as external environment.

8 Overall software architecture

8.1 Background analysis

The main objective of architecting a system means structuring it in the best way for achieving the goals specified in the requirements and specifications. Architecture defines a comprehensive view of the system where components are clearly identified.

The points defined by software architecture are:

- Component interfaces
- Component communication and dependencies
- Component responsibilities

A software architecture provides a single instantiation of a given solution and a special-purpose solution. The main characteristics are the reusable assembly of components and connectors, the usually domain-agnostic solution and high-level interaction patterns.

It's important here the highlight of the framework concept as a universal and reusable platform that provides generic functionality.

At this point we have to differentiate from components and connectors:

- Components
  - Encapsulate related functions
  - Encapsulate related data
- Connectors
  - Model interactions among components
  - Separate computation from interaction
  - Minimize component interdependencies
  - Support software evolution

From the adaptability point of view, we can gather the components, connectors and configurations analysis:

- Components
  - Give each component a single, clearly defined purpose
  - Minimize components interdependencies
  - Avoid burdening components with interaction responsibilities
  - Separate processing from data
  - Separate data from meta-data
- Connectors
  - Give each connector a clearly defined responsibility
  - Make the connector flexible
  - Support connector composability
  - Be aware of differences between direct and indirect dependencies
- Configurations
  - Leverage explicit connectors
  - Try to make distribution transparent

8.2 Architectural styles

8.2.1 Pipe and filter

A very simple, yet powerful architecture, that is also very robust. It consists of any number of components (filters) that transform or filter data, before passing it on via connectors (pipes) to other components. The filters are all working *at the same time*. The architecture is often used as a simple sequence, but it may also be used for very complex structures.

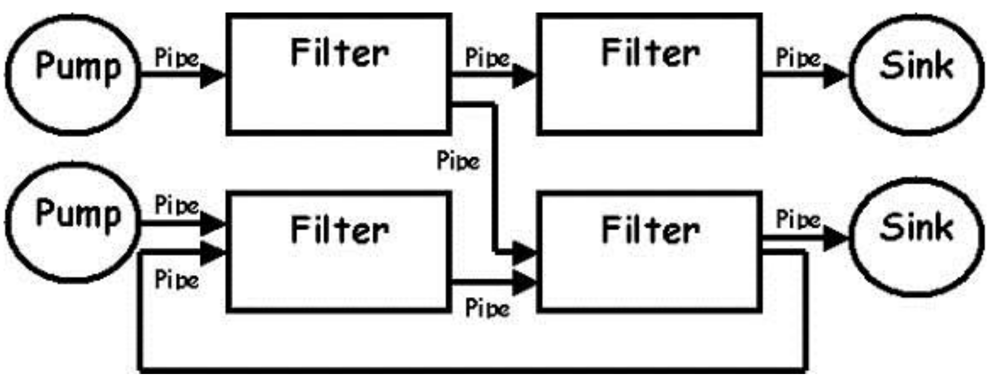

Figure 24 : Example of Pipe and Filter architecture

The filter transforms or filters the data it receives via the pipes with which it is connected. A filter can have any number of input pipes and any number of output pipes.

The pipe is the connector that passes data from one filter to the next. It is a directional stream of data, which is usually implemented by a data buffer to store all data, until the next filter has time to process it.

The pump or producer is the data source. It can be a static text file, or a keyboard input device, continuously creating new data.

The sink or consumer is the data target. It can be another file, a database, or a computer screen.

8.2.2 Blackboard

The blackboard model is a problem solving model that separates domain knowledge into heterogeneous modules called knowledge sources. A control component provides mechanisms to activate knowledge sources at the right time in order to ensure elective cooperation between them. Selection of the best enabled knowledge source is based on specific criteria and generally demands very precise tuning. In most blackboard systems, these criteria cannot be changed dynamically. Changing these criteria at run-time is however necessary in many applications like robot motion in uncertain environment, aircraft pilot advising, process control, or intensive care monitoring. These applications require highly adaptive systems that are able to adapt their meta-control strategies to dynamic configuration of demands, opportunities, and resources for behavior.

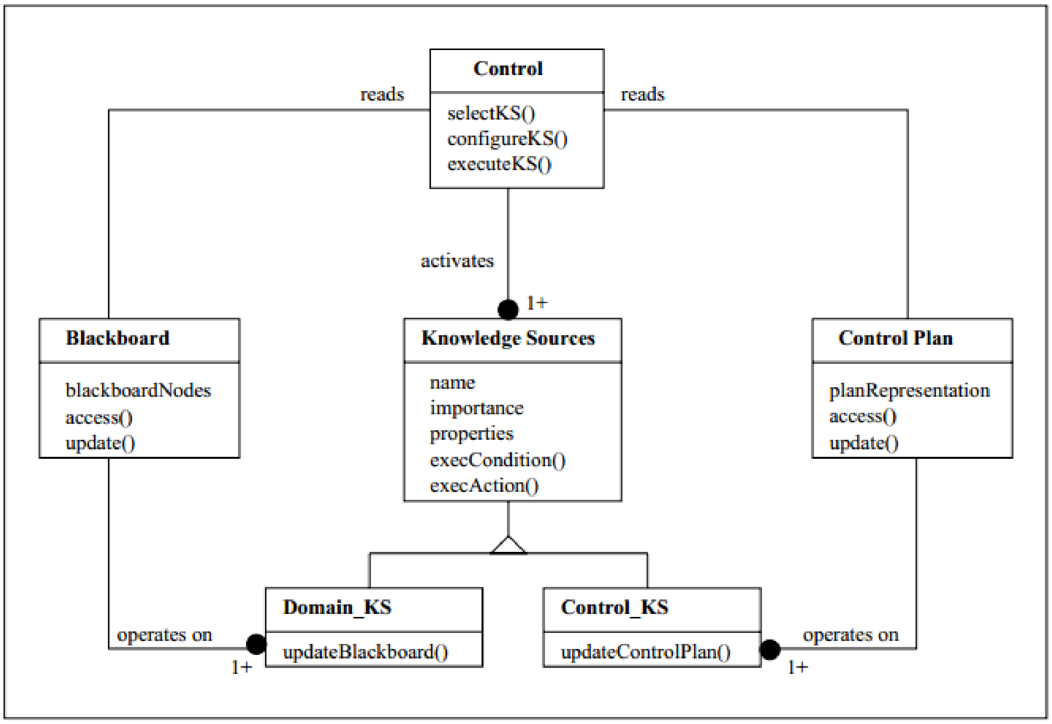

Figure 25 : Example of Blackboard architecture

8.2.3 Rule-based

The most common architecture used in expert system is Rule based system architecture.

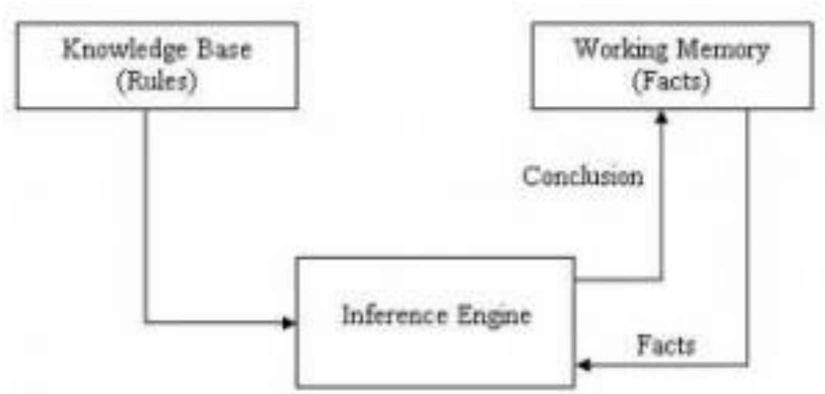

Figure 26 : Example of Rule-based architecture

As well as it's one of the most common approaches for expert systems and self-learning environments, its high capabilities for representing the adaptivity are reflected in the knowledge base with its assumptions for the adaptation to the changing environmental context information.

The following are its key modules:

1. Knowledge Base: It contains facts & rules about some specialized knowledge domain.
2. Inference Process: It accepts user input query & response to question through I/O interface. The process is carried out in 3 stages:

   - Match
   - Select
   - Execute

   During match stage the content in working memory are compared to facts & rules in knowledge base then consistent matches are in conflict set & to find an appropriate & consistent match, substitution may be required.
   Once the entire match rules have been added to conflict set during given cycle then one of rule is selected for execution.

3. Explanation Module: The explanation module provides the user with an explanation of reasoning process when requested. This is done in response to HOW & WHY query
4. I/O Interface: The I/O interface permits the user to communicate with system in more natural way by permitting the use of simple selection menu's or the use of restricted language.

8.2.4 Publish-subscribe

The publish–subscribe architectural style is a messaging pattern where senders of messages, called publishers, do not program the messages to be sent directly to specific receivers, called subscribers. Instead, published messages are characterized into classes, without knowledge of what, if any, subscribers there may be. Similarly, subscribers express interest in one or more classes, and only receive messages that are of interest, without knowledge of what, if any, publishers there are.

Publish-subscribe is a sibling of the message queue paradigm, and is typically one part of a larger message-oriented middleware system. Most messaging systems support both the pub/sub and message queue models in their API, e.g. Java Message Service (JMS).

This pattern provides greater network scalability and a more dynamic network topology.

As advantages we can mention:

- Loose coupling: Publishers are loosely coupled to subscribers, and need not even know of their existence. With the topic being the focus, publishers and subscribers are allowed to remain ignorant of system topology. Each can continue to operate normally regardless of the other. In the traditional tightly coupled client–server paradigm, the client cannot post messages to the server while the server process is not running, nor can the server receive messages unless the client is running. Many pub/sub systems decouple not only the locations of the publishers and subscribers, but also decouple them temporally. A common strategy used by middleware analysts with such pub/sub systems is to take down a publisher to allow the subscriber to work through the backlog (a form of bandwidth throttling).

- Scalability: Pub/sub provides the opportunity for better scalability than traditional client–server, through parallel operation, message caching, tree-based or network-based routing, etc. However, in certain types of tightly coupled, high-volume enterprise environments, as systems scale up to become data centers with thousands of servers sharing the pub/sub infrastructure, current vendor systems often lose this benefit; scalability for pub/sub products under high load in these contexts is a research challenge.

As disadvantages, the most serious problems with pub/sub systems are a side-effect of their main advantage: the decoupling of publisher from subscriber. A pub/sub system must be designed carefully to be able to provide stronger system properties that a particular application might require, such as assured delivery.

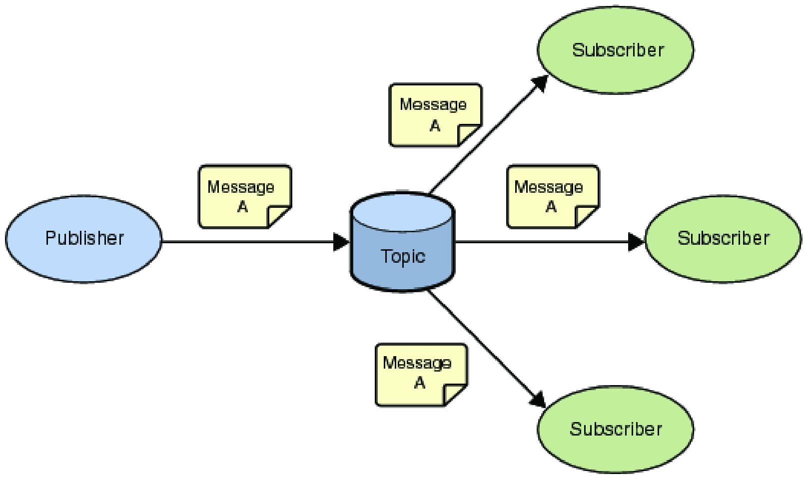

Figure 27 : Example of Publish - subscribe architecture

8.2.5 Service oriented

A service-oriented architecture (SOA) is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.

A service-oriented architecture is the underlying structure supporting communications between services. SOA defines how two computing entities, such as programs, interact in such a way as to enable one entity to perform a unit of work on behalf of another entity. Service interactions are defined using a description language. Each interaction is self-contained and loosely coupled, so that each interaction is independent of any other interaction.

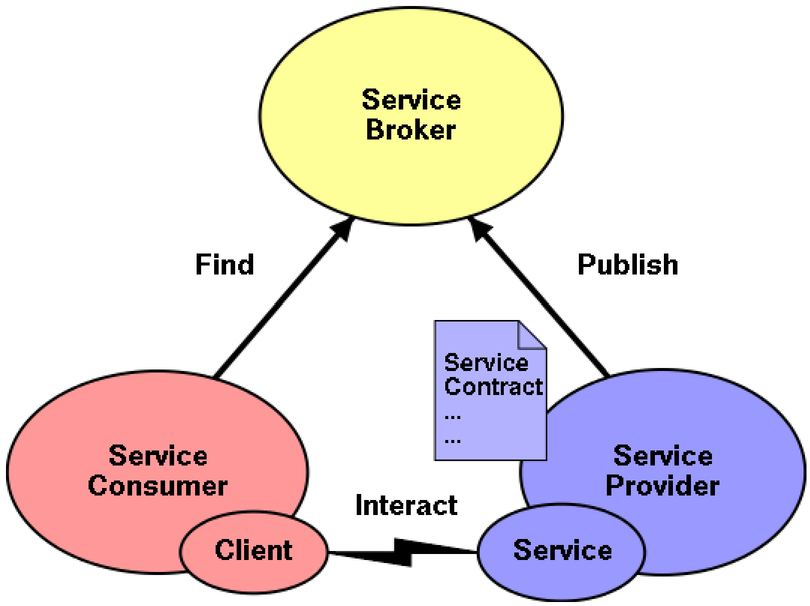

Figure 28 : Example of Service Oriented Architecture

As well as the services are static, that means that the implementation of a services can't be modified at run-time, the adaptivity in this approach is a difficult goal to achieve without of the combination of other architectural and developing techniques and approaches.

8.2.6 REST

Representational state transfer (REST) is an abstraction of the architecture of the World Wide Web. More precisely, REST is an architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements.

The main elements of a REST architecture are:

- Components: A component is an abstract unit of software instructions and internal state that provides a transformation of data via its interface.
- Connectors: A connector is an abstract mechanism that mediates communication, coordination, or cooperation among components.
- Data: Data is an element of information that is transferred from a component, or received by a component, via a connector.

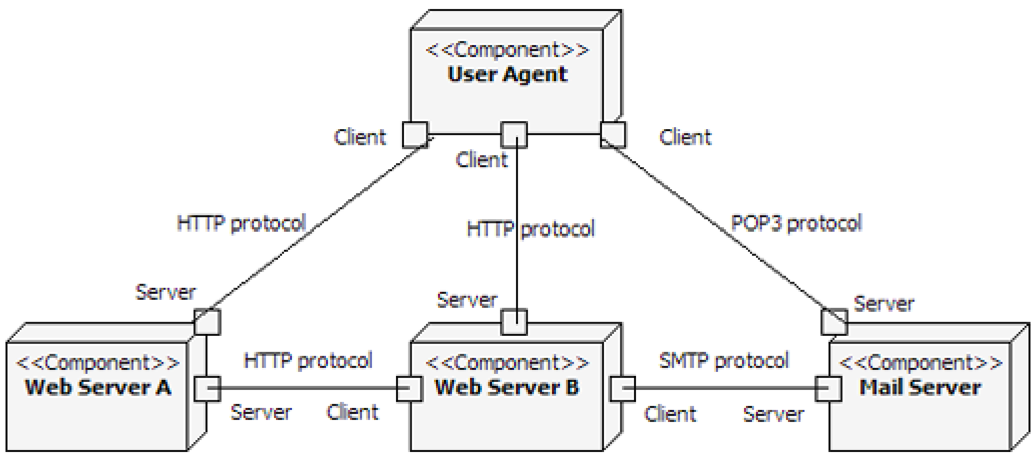

Figure 29 : Example of REST architecture

The adaptivity in this approach is given by the nature of each component, due to the respective needs of the agents depending of the context information.

8.3 A look at the research

The research works regarding software architectures on adaptive systems and previously mentioned in chapter 3.1 can be divided in two phases:

The first phase (from 1991 to 1999) covers the software architecture as a design-time tool for systems that need to be adaptive, in: [52], [53], [54] and [55].

During the second phase (2001-2007), a software architecture for self-adaptive systems concepts and paradigms has been highly developed and detailed in [56], [57], [58], [59] and [60].

8.4 Proposed methodology

Representational State Tranfer (REST) will be the architectural software technique adopted by the RTP, and consequently, adopted by the Framework for Adaptation due to its multiple benefits:

- Stateless: the server doesn't "remember" who has been making calls to it.
- Client/server architecture
- Lightweight alternative to SOAP
-

More information regarding this proposal can be found in D1.3.

The AdCOS developments have to be supported by a software suite of interoperable tools, capable of handling the following challenges:

- Offline developments: the systems onto which the AdCOS will be deployed, such as control rooms or passenger vehicles equipped with complex hardware, will not be available easily everywhere and any time. Hence the need for development environments capable to be used for offline work while preserving the capability to port the software onto real systems easily.
  This challenge relies on simulation techniques or in the capability to playback various data streams taking part in the AdCOS offline (for functions which don't require a closed-loop control).

- Real-time operation capability: the developed AdCOS systems will be required to operate in real-time. So they need to be particularly optimized.
- Multi-modality: the AdCOS software suite will have to operate multiple heterogeneous data streams which will have to be acquired, processed, fused, displayed, recorded, generated…
- Distribution: some AdCOS will have to be deployed of remote machines with operator communication and inter-systems data streaming.
- GUI management: the AdCOS may contain various display devices to interact with operators such as tactile displays or display walls.

The following set of interoperable software tools are proposed as a complete toolset for such AdCOS developments.

We will present briefly the various tools individually, then provide a description of the full proposed AdCOS development software suite.

8.5 ProSIVIC by CIVITEC

CIVITEC is a partner of HoliDes and develops the ProSIVIC simulator.

ProSIVIC is particularly suited for multi-sensor systems simulation in 3D environments. It allows simulating custom scenarios involving environment conditions, multiple sensors such as cameras, radars, laser scanners, IMUs, etc.

ProSIVIC can be configured to work in virtual time (as fast as possible, whatever time it takes to compute the sensor models rendering, dynamic models, etc.) or in real time (provided the computer is powerful enough compared to the simulation complexity) like for applications with humans in the loop.

The various data streams generated by the simulator can be accessed from other software via shared memory or network communications and using a dedicated API.

Figure 30 : The ProSIVIC simultator

8.6 RTMaps by INTEMPORA

INTEMPORA is a partner of HoliDes and develops the RTMaps software, a rapid and modular development environment for real-time applications handling multiple heterogeneous data streams. It has capability to support many data sources (such as video cameras, GPS, CAN bus, audio, motion capture, 3D sensors, DAQ, IMUs, laser scanners, radars, eye trackers and biometrics sensors, etc.)
RTMaps provides accurate time stamping for each and every data sample entering the application and operates as a multi-threaded environment to be able to manage different data streams with different frame rates, including event-based sources.

It is capable of recording and playing back any kind of data streams in a synchronized manner.

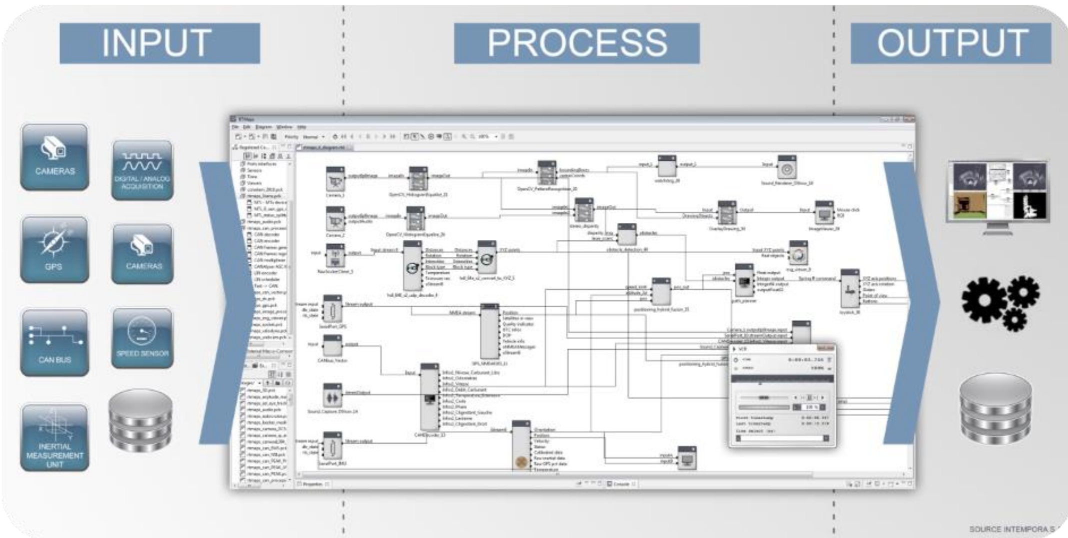

Figure 31 : The RTMaps Studio

RTMaps is particularly suited for the following applications:

- Perception thanks to multi-sensor data fusion
- Multimodal HMIs development
- After Action Analysis of operator / driver / pilot behavior, including in distributed environment with cooperating operators.

8.7 Matlab/Simulink & dSPACE

Simulink is a component based environment operated on a synchronous, time-step execution mode and developed by The Mathworks.
It is particularly suited to simulate dynamic models and to develop command-control laws.

Once a Simulink model has been designed, it can be tested in simulation or generated to C code towards various real-time execution environments. The dSPACE AutoBox is an ECU emulator with hard-real-time operation capability and easy integration with Simulink. It is widely spread in the automotive domain for development of control functions.

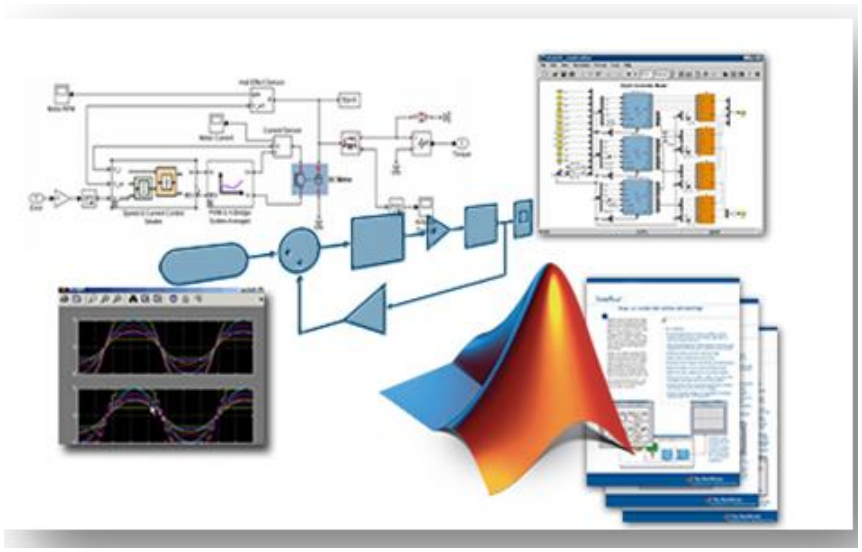

Figure 32 : Matlab Simulink

8.8 Qt / QML/ QtCreator

Qt is an open-source cross-platform C++ API for powerful graphical user interfaces.

QML is a high-level scripting language, based on Qt, which allows to develop nice GUIs without having to get into C++. QML is easily evolutive and can be used by non-programmers.

Qt Creator is a cross-platform C++, JavaScript and QML integrated development environment which is part of Qt framework.

Figure 33 : QtCreator, a graphical editor for Qt and QML GUIs

8.9 An AdCOS development tool chain

The software tools presented above have been interfaced together to be able to inter-operate in real-time and to cover the maximum range of the AdCOS development steps:

- Offline developments, tests, validation
- Online real-time operation in standalone or distributed mode

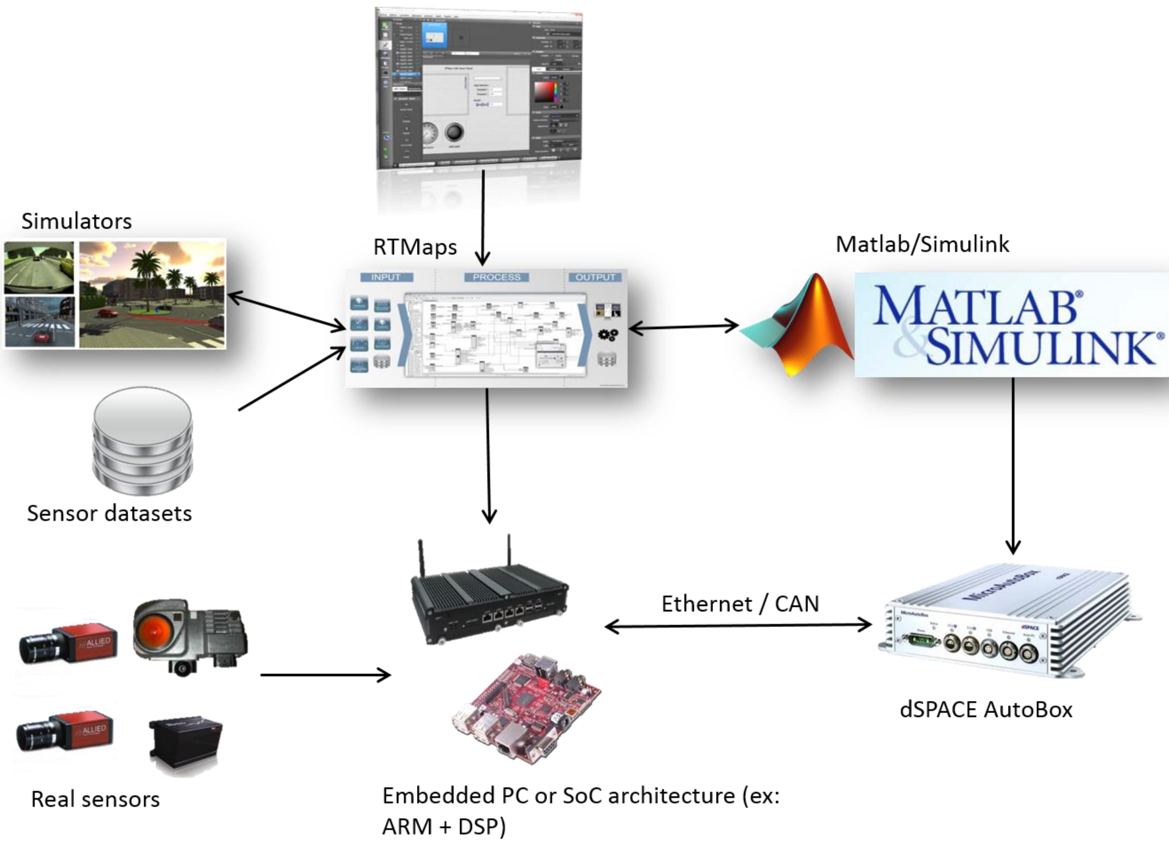

Figure 34 : An AdCOS development tool chain

Figure 34 presents the workflow proposed for the AdCOS development tool chain.

The upper part presents the tool chain for offline developments in a simulator environment or via sensors data playback functions, while the lower part presents the porting of the developed applications onto a real prototype.

The left column presents the sensors & actuators interface (either simulated, recorded & played back, or interfaced in real-time).

The column in the middle presents RTMaps, in charge of integration and execution of high-level functions such as data acquisition, image processing, signal processing, decision & data visualization (eventually via Qt / QML).
And the column on the right presents command-control laws which can be run either in co-simulation (offline) in Simulink, or in real-time on a dSPACE target for instance.

This architecture will be used, among others, in WP9 in the CRF demonstrator and by IFSTTAR for the COSMO-SIVIC simulator.
Here are some example applications:

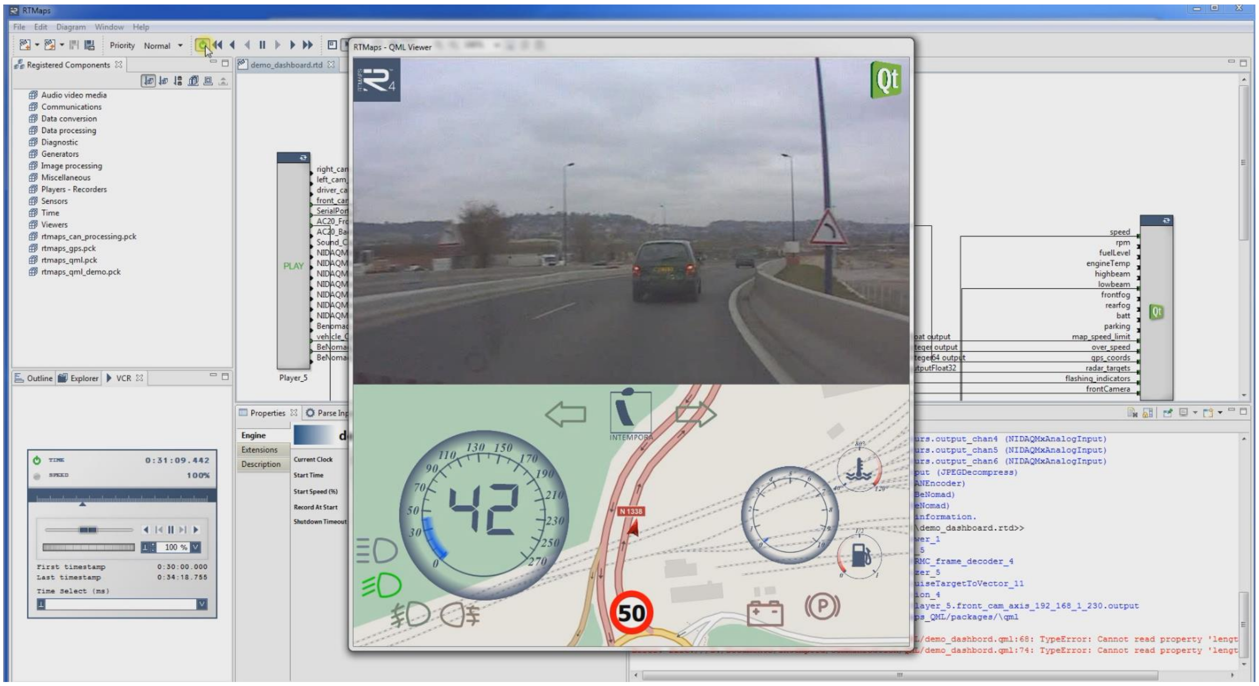

Figure 35 : QML graphical user interface

Figure 35 presents a QML GUI executed and animated with an RTMaps diagram in playback mode (front video, GPS, CAN bus).

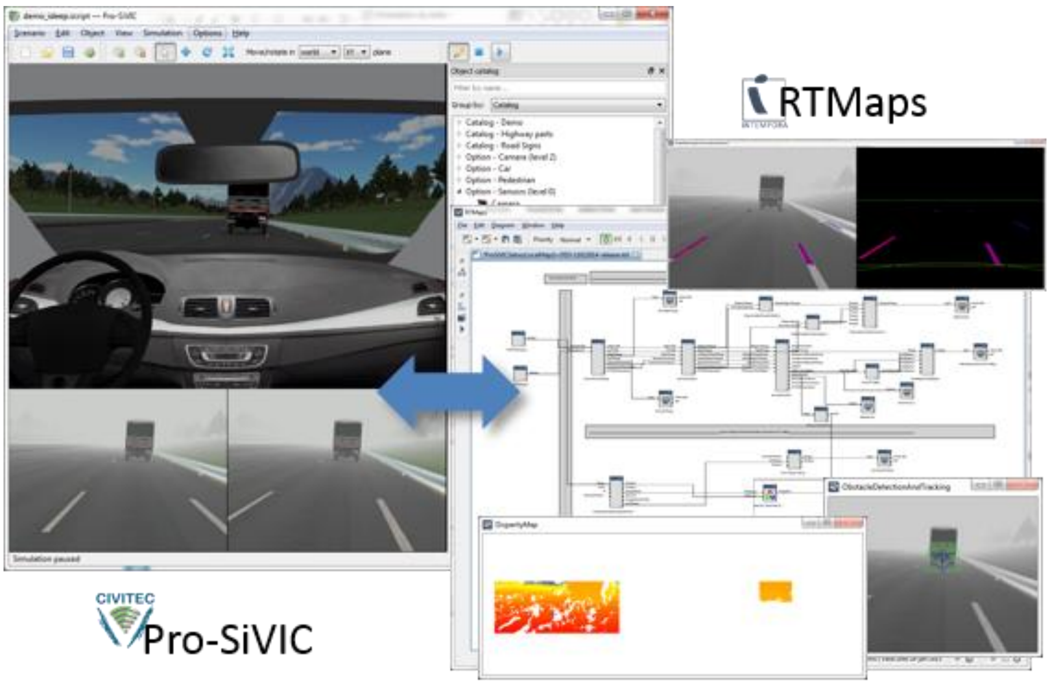

Figure 36 : Stereovision algorithm in RTMaps

Figure 36 presents a stereovision algorithm and a lane marking detector (from IFSTTAR / LIVIC) executed in RTMaps and fed by synthetic sensor data generated in the ProSIVIC simulator (here a stereovision camera with simulated fog).

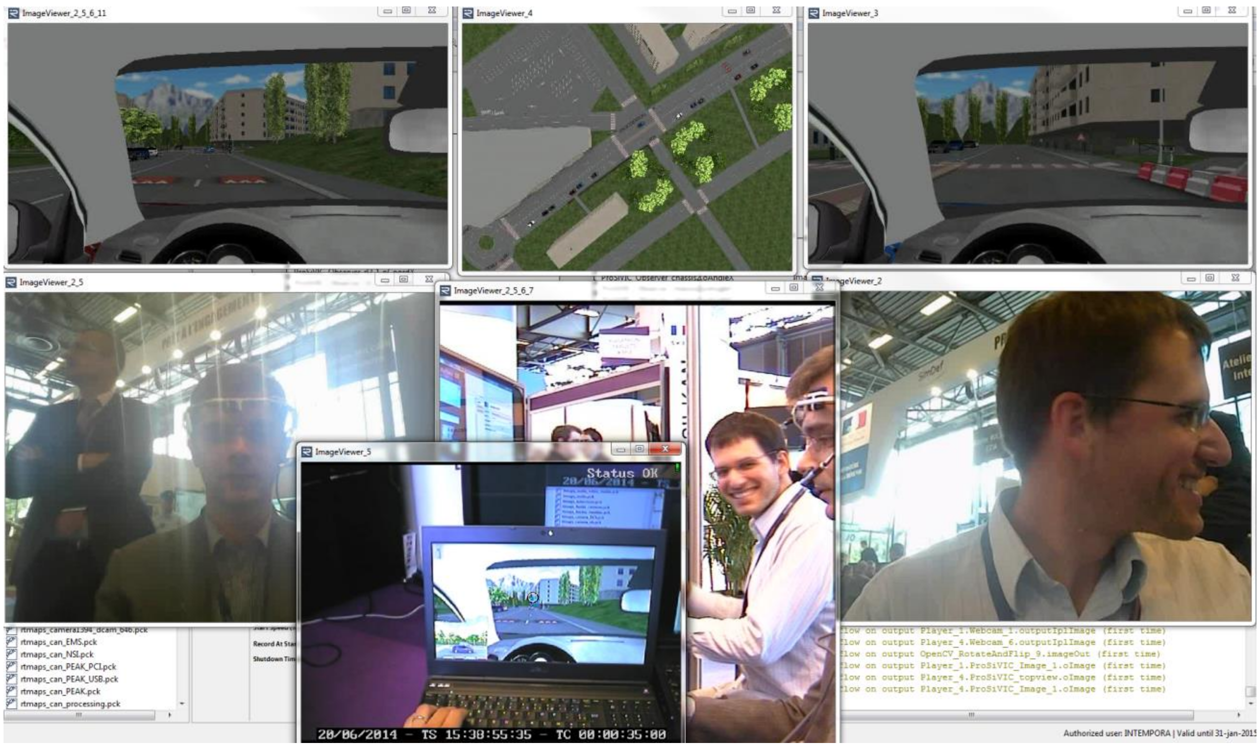

Figure 37 : Synchronized real-time playback application

Figure 37 presents a synchronized real-time playback application of a distributed simulator system for debriefing and After Action Analysis (AAA) applications.

The driving simulator (here ProSIVIC) was distributed on two laptops.
The recording was performed locally on each laptop but using a synchronized clock for time stamping. Recorded signals in RTMaps were:

- Driver view (simulated)
- Bird view (simulated)
- Drivers laptop webcams
- Scene view from an IP camera (AXIS)
- Audio stream
- Eye tracker for one of the drivers (Pertech)

The recorded datasets are then gathered on a single computer for synchronized playback and behavioral analysis.

9 Specific role of some partners

Role of CRF

CRF will develop a distraction classifier (estimation of both cognitive and visual distraction) of a car-driver in WP2, together with UTO partner. In case of a distracted driver, the AdCoS may either inform the driver in order to allow him to refocus on the driving task (actually, this opportunity is still to be decided at HMI and strategy level), or it may change the strategies of the applications implemented in the CRF demonstrator, on which specific ADAS will be developed, such as:

- Lane-change Assistant (LCA)
- Extended Forward collision Warning (e-FCW), which include warning and emergency autonomous braking
- Overtaking Assistant (OA)

A more complete description of these functions, the list of the addressed requirements (with respect to D9.1), the system architecture and specifications will be detailed in the WP9 deliverable D9.3.

CRF will contribute to the design of the WP3 adaptation framework taking these points into account.

In addition, CRF will contribute to the adaptation module developing a specific driver model, which includes a kind of virtual driver, named **co-pilot**, again together with UTO, and a **driver's intention** estimator, together with OFF partner. The first is based on Markovian Decision Process (MDP), while the second on Bayesian Neural Networks (BNN).

Both the prediction of driver's intention and the classification of driver's cognitive state are inputs to the co-pilot, which becomes in this way the core of adaptive systems (namely, the AdCoS). In fact, the three aforementioned applications (that is, LCA, OA and e-FCW) can adapt themselves to:

- Predicted manoeuvre in the near future (the intention of the driver).
- Distraction of the driver
- Risk and criticality of the external situation

These modules and framework for adaptation will be used implemented and integrated on the prototype vehicle using RT-Maps.

Role of IFS

In the frame of HoliDess project as a whole, IFSTTAR will develop (in WP2) a COgnitive Simulation MOdel of the car DRIVEr (named COSMODRIVE) to be interfaced (in WP4) with 2 tools, that are Pro-SIVIC (provided by CIVITEC) and RT-MAPS (coming from INTEMPORA), with the aim to have a *Virtual Human Centred Design (V-HCD) platform* of AdCoS (named COSMO-SIVIC), to be used in WP9 (in Automotive domain), in order to virtually design and test future AdCoS. At last, the challenge is indeed to have a V-HCD platform integrating (1) a human driver model (i.e. COSMODRIVE) using a "virtual eye" for road scene scanning, and able to drive (2) a virtual car (3) equipped with virtual ADCOS, for progressing in a virtual 3-Dimensional environment.

The following figure provides an overview of this future COSMO-SIVIC "Virtual HCD platform" (as one of the HF-RTP demonstrator in WP9) to be developed by IFSTTAR during the project:

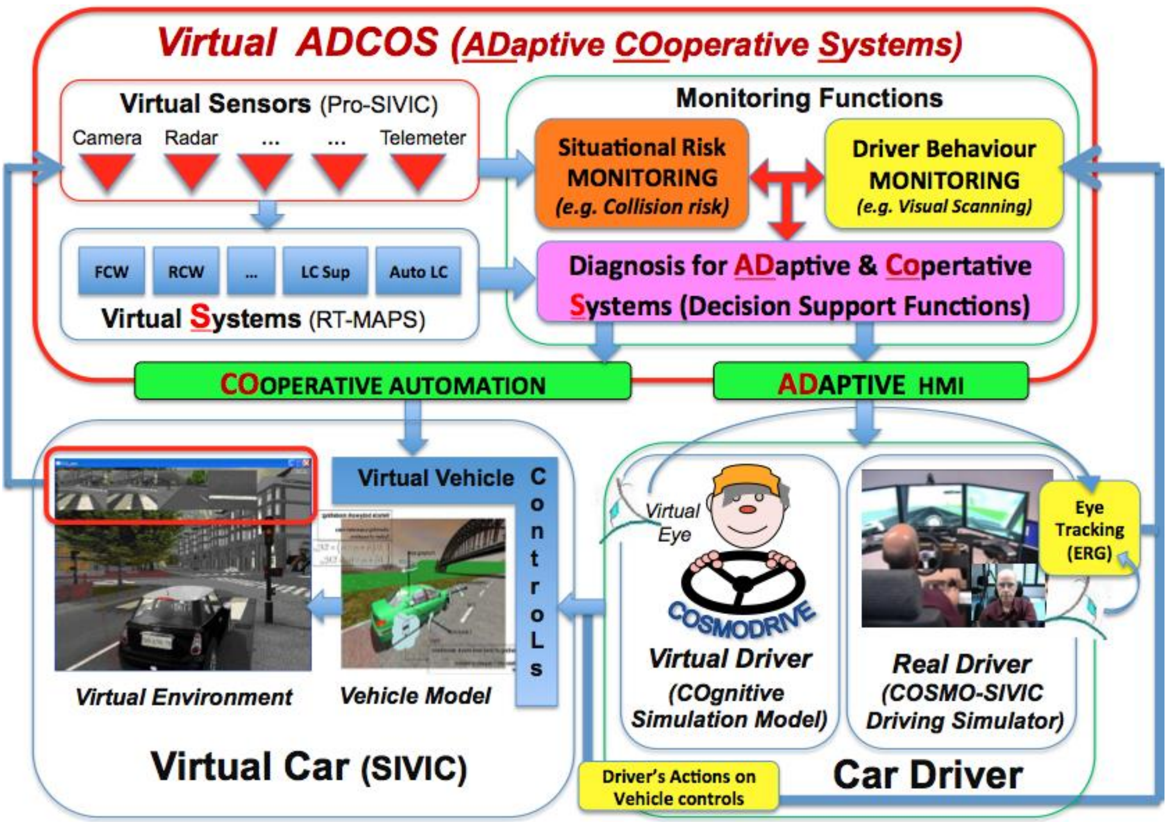

Figure 38 : Overview of the IFSTTAR "Virtual HCD platform" (COSMODRIVE + Pro-SIVIC + RT-MAPS)

In the specific frame of the WP3, IFSTTAR will design and develop a set of Monitoring and Decision Support Functions, to be integrated in the AdCos for supporting Adaptive and Cooperative abilities of driving Aids, according to human drivers' errors (due to visual distraction simulated with COSMODRIVE, or observed among real drivers on IFS driving simulator), and to the Situational Risk (e.g. collision risk with a car ahead or with a car in rear/lateral position). As input, these Monitoring Functions will take into account, from one side, drivers' behaviours (i.e. actions on vehicle pedals and steering wheel, and visual scanning assessed through eye tacking measures; a collaboration with ERGONEERS is under discussion regarding this specific issues) and, from the other side, situational data collected through sensors (virtually simulated on the V-HCD with Pro-SIVIC software) in order to assess the criticality of the traffic situation. By combining these two types of Monitoring Functions (respectively focused on the Driver and the traffic Situation), diagnosis algorithms will be developed by IFSTTAR in order to assess (1) drivers behaviour adequacy and (2) collision risks. Then, these diagnoses will be used in the virtual ADCOS as decision support rules for adapting driving aids in a cooperative way (e.g. adapt H-M Interaction modalities, active or not driving aids based on vehicle automation), according to the driver's status (e.g. visually distracted), behavioural errors (e.g. inadequate or risky manoeuvre implemented), and to the collision risks with the other vehicles currently interacting with the ego-car.

Finally, in the frame of WP9, this integrative COSMO-SIVIC "V-HCD platform" will be used to simulate driving performance of a human driver with or without driving aids (from normal behaviours to critical behaviour due to visual distraction) in order to support AdCos virtual design process at 2 levels. At the earliest design stages, COSMODRIVE will be used to estimate human drivers' performances in case of unassisted driving, in order to identify critical driving scenarios for which a given AdCoS should be provided for supporting the driver and to specify it in an ergonomics way. Then, after virtual AdCOS development, it will be possible to virtually assess its *effectiveness* for different variations of the critical scenarios previously identified, and to check its *efficiency* according to human drivers' needs (through COSMODRIVE-based simulation).

Role of TWT

TWT will develop a cognitive model of a car driver in WP2 that focusses on auditory input (audio scene within the car) and visual input (facial expressions of the driver). From the output of this model, driver distraction can be estimated. In case of a distracted driver, the AdCoS may either inform the driver in order to allow him to refocus on the driving task, or it may signal this information to other system components (such as ADAS). This scenario is depicted in the following image:

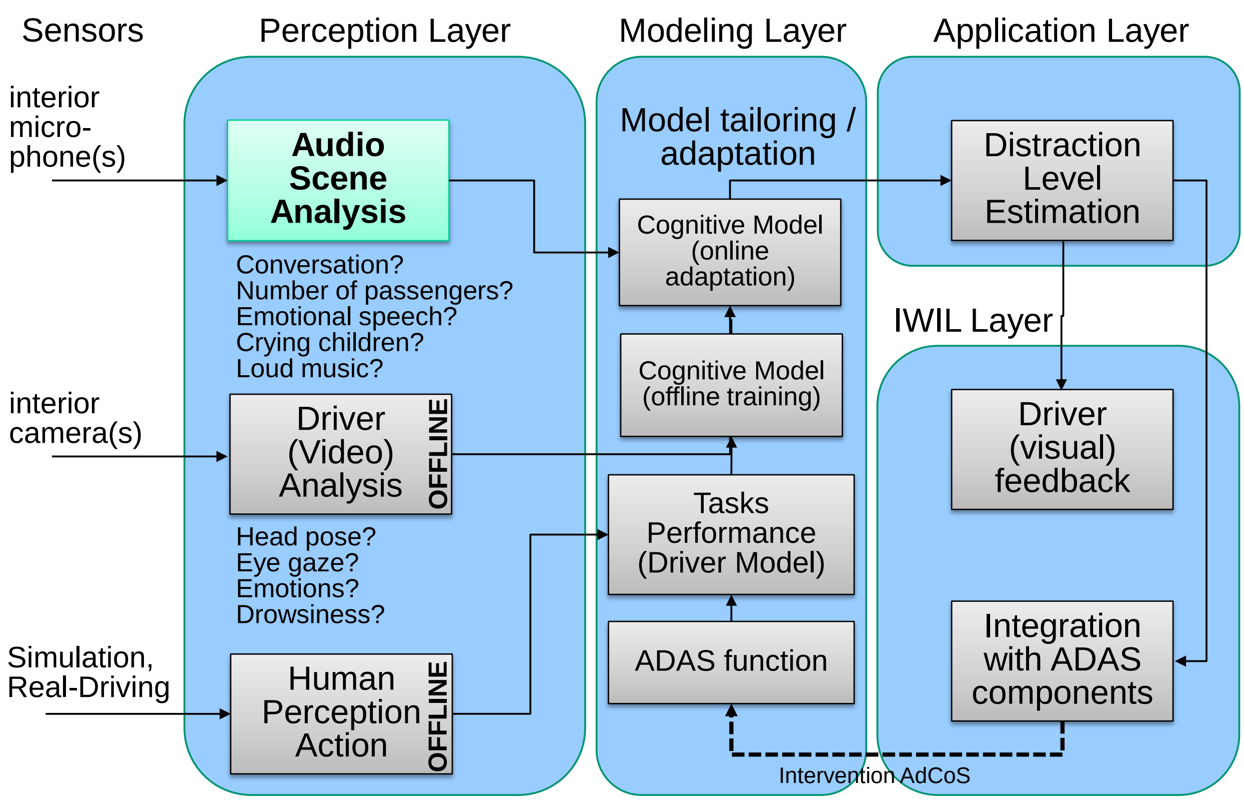

Figure 39: Perception, modelling and application layers in driver assistance

From this scenario, the following requirements for adaptivity were deduced (cf. D3.1):

- WP9_TWT_AUT_REQ04 – The driver should be informed about his/her distraction level in order to prevent further distraction (*"adaptation" of the human driver*).
- WP9_TWT_AUT_REQ07/08 – Control measurements with eye-tracking and/or analysis of facial expressions (*adaptation of cognitive model*).

- WP9_TWT_AUT_REQ13 – Driver related (observations of driver) and driving related information (e.g., lane-keeping) can implicitly tag important events and helps learning *structured models that adapt to individual driver's profiles.*
- WP9_TWT_AUT_REQ14 – The estimated distraction level can be used to *adapt functions and/or thresholds of other systems* (e.g., ADAS).

TWT will contribute to the design of the WP3 adaptation framework with those requirements in mind.

In addition, TWT will contribute an adaptation module based on audio-based driver distraction estimation. This includes the definition of the needed input data and the various data that is made available by the developed model. Potential adaptation tasks include:

- Adaptivity to external noise (e.g., open window, different road types)
- Adaptivity to number of talking participants
- Adaptivity to augmented sensor modalities (audio signals, sensor fusion models, insights from WP2 technical and human cognitive models)
- Model adaptivity to conversational context (who is speaking to whom – interaction with cooperativity issues, adaptive tracking of Audio-based identities)

After the design of the WP3 adaptation framework, its modules will be used to detail the necessary execution steps for the scenario described above. In addition, its implementation and integration in RTMaps will be considered.

10 Conclusion

The framework for adaptation that has been evaluated in this document is compatible with the AdCos from the project partners coming from different industries. Good examples from the partners underline the reasonability of the formulated framework.

11 References

[1] ABOWD, G. D., DEY, A. K., BROWN, P. J., DAVIES, N., SMITH, M., AND STEGGLES, P. Towards a better understanding of context and context-awareness. In Handheld and ubiquitous computing (1999), Springer, p. 304–307.

[2] BOLCHINI, C., CURINO, C. A., QUINTARELLI, E., SCHREIBER, F. A., AND TANCA, L. A data-oriented survey of context models. SIGMOD Rec. 36, 4 (Dec. 2007), 19–26.

[3] BOZDAG, E. A Survey of Extensions to the Contract Net Protocol.

[4] DOBSON, S. et al., "A Survey of Autonomic Communications", ACM Transactions on Autonomous and Adaptive Systems (TAAS), pages: 223–259, volume 1, issue 2, December 2006.

[5] DOWLING, J. The decentralized coordination of self-adaptive components for autonomic distributed systems. PhD thesis, Cite-seer, 2004.

[6] DURFEE, E. H., ORTIZ JR, C. L., AND WOLVERTON, M. J. A survey of research in distributed, continual planning. AI Magazine 20, 4 (1999), 13.

[7] ENDRISS, U. Monotonic concession protocols for multilateral negotiation. In Proceedings of the fifth international joint conference on Autonomous agents and multi-agent systems (2006), ACM, p. 392–399.

[8] ENDSLEY, M. R.(1995a) Measurement of situation awareness in dynamic systems. Human Factors: The Journal of the Human Factors and Ergonomics Society 37, 1 (1995), 65–84.

[9] EROL, K., HENDLER, J. A., AND NAU, D. S. UMCP: a sound and

complete procedure for hierarchical task-network planning. In AIPS (1994), vol. 94, p. 249–254.

[10] FEIGH, K. M., DORNEICH, M. C., AND HAYES, C. C. To-ward a characterization of adaptive systems a framework for re-searchers and system designers. Human Factors: The Journal of the Human Factors and Ergonomics Society 54, 6 (2012), 1008–1024.

[11] GERKEY, B. P., MATARIC, M. J. A formal analysis and taxonomy of task allocation in multi-robot systems. The International Journal of Robotics Research 23, 9 (2004), 939–954.

[12] GAT, E., "Three-layer Architectures", Artificial Intelligence and Mobile Robots, MIT/AAAI Press, Cambridge, MA, USA pages 195-210.

[13] KOMENDA, A., VOKRINEK, J., PECHOUCEK, M., WICKLER, G., DALTON, J., AND TATE, A. I-globe: Distributed planning and coordination of mixed initiative activities. Proceedings of Knowledge Systems for Coalition Operations (KSCO 2009).

[14] LABORIE, P. Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. Artificial Intelligence 143, 2 (Feb. 2003), 151–188.

[15] LABORIE, P., AND GHALLAB, M. Planning with sharable resource constraints. In IJCAI (1995), vol. 95, p. 1643.

[16] LUDTKE, A. HoliDes. Full project proposal addressing ASP8, 2013.

[17] MARIE DESJARDINS, M. W. Coordinating planning activity and information flow in a distributed planning system.

[18] MORANDINI, M., PENSERINI, L., AND PERINI, A. Towards goal-oriented development of self-adaptive systems. SEAMS '08 Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems. Pages 9-16

[19] MÜLLER-SCHLOER, C. and SCHMECK, H., "Organic Computing: A Grand Challenge for Mastering Complex Systems", it - Information Technology, pages 135–141, volume 52, issue 3, 2010.

[20] OBERSCHELP, O. et al., "Design of self-optimizing agent-based controllers”, In Proceedings of the Workshop 2002 - Agent-Based Simulation 3, Passau, Germany, April 2002.

[21] PROTHMANN, H., "Organic traffic control", Karlsruhe Institute of Technology, ISBN 978-3-86644-725-7, 2011.

[22] SALEHIE, M., AND TAHVILDARI, L. Self-adaptive software: Landscape and research challenges. ACM Trans. Auton. Adapt. Syst. 4, 2 (May 2009), 14:1–14:42.

[23] SMITH, D. E., FRANK, J., AND JÓNSSON, A. K. Bridging the gap between planning and scheduling. The Knowledge Engineering Review 15, 01 (2000), 47–83.

[24] SUBRAMANIAN, N., AND CHUNG, L. Software architecture adaptability: an NFR approach. In Proceedings of the 4th Inter-national Workshop on Principles of Software Evolution (2001), ACM, p. 52–61.

[25] WOOLDRIDGE, M. An Introduction to Multi-Agent Systems. John Wiley & Sons, June 2009.

[26] ZLOT, R., AND STENTZ, A. Complex task allocation for multiple robots. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005 (Apr. 2005), pp. 1515–1522.

[27] ACM Transactions on Autonomous and Adaptive Systems (TAAS). Volume 9 Issue 1. March 2014. Article No 2. NY; USA

[28] ESFAHANI N., ELKHODARYA., AND MALEK S*. A Learning-Based Framework for Engineering feature-oriented self-adaptive software Systems. July 2013.*

[29] GUINEA, SAM. Software architecture for adaptation systems. Politecnico di Milano. November 2013

[30] Designing Dynamic Distributed Cooperative Human-Machine-Systems(D3CoS) www.d3cos.eu/index.php/objectives

[31] Highly automated vehicles for intelligent transport (HAVE-IT), www.haveit-eu.org

[32] Cognitive Adaptive Man-Machine Interface (CAMMI), www.cammi.eu

[33] Pro-active decision support for data-intensive environments. www.astute-project.eu

[34] BERNON, C., GLEIZES, M. P., PEYRUQUEOU, S., AND PICARD, G. 2003. Adelfe: A methodology for adaptive multi-agent systems engineering. In ESAW. Lecture Notes in Computer Science, vol. 2577. Springer, 156–169.

[35] BERNON, C., CAMPS, V., GLEIZES, M. P., AND PICARD, G. 2004. Designing agents' behaviors and interactions within the framework of adelfe methodology. In ESAW. Lecture Notes in Computer Science, vol. 3071. Springer, 311–327.

[36] DEY, A. K. 2001. Understanding and Using Context. Personal Ubiquitous Computing, Vol. 5, No. 1, pp. 4-7.

[37] ENDSLEY, M.R. (1995b). Toward a theory of situation awareness in dynamic systems. Human Factors 37(1), 32–64.

[38] FIKSEL, J. (2003) 'Designing resilient, sustainable systems', in Environmental Science and Technology, Vol. 37, No. 23

[39] FORLIZZI J. AND DISALVO C. 2006. Service robots in the domestic environment: a study of the roomba vacuum in the home. In Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction (HRI '06). ACM, New York, NY, USA, 258-265.

[40] GLIZE, P : L'Adaptation des Systemes a Fonctionnalité Emergente par Auto-Organisation Cooperative. Thèse de doctorat, Université Paul Sabatier - Toulouse III, 2001.

[41] HOLLNAGEL, D.D. WOODS AND N. LEVESON, EDS. "Resilience Engineering: Concepts and Precepts" (pdf). Ashgate, Aldershot, UK, 2006.

[42] R. PARASURAMAN, T.B. SHERIDAN, AND C.D. WICKENS, "A model for types and levels of human interaction with automation", presented at IEEE Transactions on Systems, Man, and Cybernetics, Part A, 2000, pp.286-297.

[43] SCERBO, M.W. (2001). Adaptive automation. In W. Karwowski (Ed.), International encyclopedia of ergonomics and human factors (pp. 1077-1079). London: Taylor and Francis, Inc.

[44] WEISER, M. The computer for the twenty-first century. Sci. Am. 1991, 3, 94-104.

[45] WREATHALL, J. 'Properties of resilient organizations: an initial view', in Hollnagel, E., Woods, D.D. and Leveson, N. (Eds.): Resilience Engineering: Concepts and Precepts, Ashgate

[46] ZIEBA, S., POLET, P., VANDERHAEGEN, F. AND DEBERNARD, S. (2009) 'Resilience of a human-robot system using adjustable autonomy and human-robot collaborative control', Int. J. Adaptive and Innovative Systems, Vol. 1, No. 1, pp.13–29.

[47] ZIEBA S., POLET P., VANDERHAGEN F., DEBERNARD S. (2010). Principles of adjustable autonomy: a framework for resilient human machine cooperation. Cognition, Technology and work, 12 (3), pp. 193-203.

[48] SILAGHI M.-C., and YOKOO M. 2009. Adopting: unifying asynchronous distributed optimization with asynchronous backtracking. Autonomous Agents and Multi-Agent Systems 19, 2, 89–123.

[49] VALCKENAERS P., HADELI K., GERMAIN B.S., VERSTRAETE P., and RUSSEL H.V.2006. Emergent short-term forecasting through ant colony engineering in coordination and control systems. Advanced Engineering Informatics 20, 3, 261–278.

[50] HEYLIGHEN F., and GERSHENSON C. 2003. The meaning of self-organization in computing. IEEE Intelligent Systems, 72–75.

[51] DI MARZO SERUGENDO G., GLEIZES M.-P. and KARAGEORGOS A. 2006. Self-Organisation and Emergence in Multi-Agent Systems: An Overview. Informatica, International Journal of Computing and Informatics, 30(1):45–54.

[52] GORLICK AND RAZOUK, (1991) "Using Weaves for Software Construction and Analysis", ICSE '91, Proceedings of the13th International Conference on Software Engineering Pages 23-34.

[53] Taylor, Medvidovic, Anderson, Whitehead, and Robbins, (1995) "A Component-and-Message-Based Architectural Style for GUI Software", ICSE '95 Proceedings of the 17th International Conference on Software Engineering, pages 295-304

[54] MAGEE, DULAY, AND KRAMER, "Regis: a Constructive Development Environment for Distributed Programs", Distributed Systems Engineering, Volume 1,Number 5, 1994, pages 304-312

[55] OREIZY, GORLICK, TAYLOR, HEIMBINGER, JOHNSON, MEDVIDOVIC, QUILICI, ROSENBLUM, AND WOLF, "An Architecture-Based Approach to Self-Adaptive Software", IEEE Intelligent Systems, Volume 14 Issue 3, May 1999, page 54-62

[56] DOWLING, J AND CAHILL, V., "The K-Component Architecture Meta-Model for Self-Adaptive Systems", 2001.Reflection 2001 LNCS 2192, pages 81-88

[57] GOMAA AND HUSSEIN, "Dynamic Software Reconfiguration in Software Product Families", Software Product-Family Engineering, 5th International Workshop, 2003, page 435-444

[58] GARLAN, CHENG, AND SCHMERL, (2003) "Increasing System Dependability through Architecture-based Self-repair", in Architecting dependable systems, pages 61-89, Springer-Verlag Berlin,

[59] HAWTHORNE, M. J., & PERRY, D. E. (2004). Exploiting Architectural Prescriptions for Self-Managing, Self-Adaptive Systems: A Position Paper.

[60] KRAMER AND MAGEE, "Self-Managed Systems: an Architectural Challenge", in Future of Software Engineering 2007 , pages 259 - 268

[61] WEYNS AND HOLVOET, "An Architectural Strategy for Self-Adapting Systems", Proceedings of SEAMS '07 Proceedings of the 2007

International Workshop on Software Engineering for Adaptive and Self-Managing Systems, page 3

[62] GEORGAS, VAN DER HOEK, AND TAYLOR, "Using Architectural Models to Manage and visualize Runtime Adaptation", IEEE Computer 2009 October (vol. 42 no. 10), pp. 52-60.

[63] BARESI, L., & GUINEA, S. (2011). A-3: An Architectural Style for Coordinating Distributed Components. In Proceedings of the 2011 Ninth Working IEEE/IFIP Conference on Software Architecture (pp. 161-170). IEEE Computer Society

[64] CHENG, GARLAN, AND SCHMERL, "Stitch: A Language for Architectural-based Self-Adaptation", Journal of Systems and Software, Special Issue on State of the Art in Self-Adaptive Systems, 2012, Pages 2860–2875

[65] DEY, A.K., NEWBERGER, A. (2009)" Support for context-aware intelligibility and control"; CHI '09 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Pages 859-868

[66] HOLLNAGEL, E. (1998). *Cognitive reliability and error analysis method: CREAM* (1st ed.). Oxford, New York: Elsevier.

[67] MILLER, C. D., & PARASURAMAN, R. (2007). Designing for Flexible Interaction Between Humans and Automation: Delegation Interfaces for Supervisory Control. *Human Factors*, *49*(1), 57–75.

[68] NORMAN, D. A. (1989). *The design of everyday things* (1st ed.). New York: Basic Books.

[69] RASMUSSEN, J., & VICENTE, K. J. (1990). Ecological interfaces: A technological imperative in high-tech systems? *International Journal of Human-Computer Interaction*, *2*(2), 93–110. doi:10.1080/10447319009525974

[70] WICKENS, C. D., HOLLANDS, J. G., BANBURY, S., & PARASURAMAN, R. (1992). *Engineering psychology and human performance* (1st ed.).