



**HoliDes**  
**Holistic Human Factors Design of**  
**Adaptive Cooperative Human-**  
**Machine Systems**



## D2.5 – Modelling Techniques and Tools V1.5

<b>Project Number:</b>	332933												
<b>Classification:</b>	Public												
<b>Work Package(s):</b>	WP2												
<b>Milestone:</b>	M3												
<b>Document Version:</b>	V1												
<b>Issue Date:</b>	30.06.2015												
<b>Document Timescale:</b>	Project Start Date: October 1, 2013												
Start of the Document:	Month 19												
Final version due:	Month 21												
<b>Deliverable Overview:</b>	<p><b>Main document:</b> Modelling Techniques and Tools V1.5 (Public)</p> <p><b>Annex I:</b> Requirements Update (confidential)</p> <p><b>Annex II:</b> Application of Task Models in HoliDes (confidential)</p> <p><b>Annex III:</b> HoliDes Integration Plan (confidential)</p> <p><b>Annex IV:</b> Human Operator Classification (confidential)</p>												
<b>Compiled by:</b>	Sillaurren, Sara – TEC												
<b>Authors:</b>	<table style="width: 100%; border: none;"> <tr> <td style="width: 50%;">Bellet, Thierry - IFS</td> <td style="width: 50%;">López, Jesús - TEC</td> </tr> <tr> <td>Thurlings, Marieke - TWT</td> <td>Magnaudet, Mathieu - ENA</td> </tr> <tr> <td>Botta, Marco - UTO</td> <td>Osterloh, Jan-Patrick - OFF</td> </tr> <tr> <td>Sillaurren, Sara - TEC</td> <td>Borchers, Svenja - TWT</td> </tr> <tr> <td>Eilers, Mark - OFF</td> <td></td> </tr> <tr> <td>Feuerstack, Sebastian - OFF</td> <td></td> </tr> </table>	Bellet, Thierry - IFS	López, Jesús - TEC	Thurlings, Marieke - TWT	Magnaudet, Mathieu - ENA	Botta, Marco - UTO	Osterloh, Jan-Patrick - OFF	Sillaurren, Sara - TEC	Borchers, Svenja - TWT	Eilers, Mark - OFF		Feuerstack, Sebastian - OFF	
Bellet, Thierry - IFS	López, Jesús - TEC												
Thurlings, Marieke - TWT	Magnaudet, Mathieu - ENA												
Botta, Marco - UTO	Osterloh, Jan-Patrick - OFF												
Sillaurren, Sara - TEC	Borchers, Svenja - TWT												
Eilers, Mark - OFF													
Feuerstack, Sebastian - OFF													
<b>Reviewers:</b>	Nico Van den Berg – UMC Landini, Elisa – REL												
<b>Technical Approval:</b>	Jens Gärtner, Airbus Group Innovations												
<b>Issue Authorisation:</b>	Sebastian Feuerstack, OFF												

© All rights reserved by HoliDes consortium

This document is supplied by the specific HoliDes work package quoted above on the express condition that it is treated as confidential to those specifically mentioned on the distribution list. No use may be made thereof other than expressly authorised by the HoliDes Project Board.



<b>RECORD OF REVISION</b>		
Date (DD.MM.JJ)	Status Description	Author
20.04.2015	Initial Structure	JPO, OFF & Sara Sillaurren, TEC
24.04.2015	Contribution to Pilot Pattern Classifier	Jesus López, TEC
07.05.2015	Contribution to djnn	Mathieu Magnaudet, ENA
11.05.2015	Contribution to Detection of Driver distraction based on car measurement	Marieke Thurlings, TWT
12.05.2015	Contribution to Driver Distraction	Marco Botta, UTO
14.05.2015	Contribution to COSMODRIVE	Thierry Bellet, IFS
14.05.2015	First integrated version with contributions	Sara Sillaurren, TEC
04.06.2015	Contribution to section 2.1.5 HMI Iteration models	Mathieu Magnaudet, ENA
04.06.2015	Contribution to section 2.1.4	Zdenek Moraveck, HON
04.06.2015	Contribution to section 2.1.4	Marieke Thurlings & Svenja Borchers, TWT
04.06.2015	Contribution to section 2.1.4	David Käthner, DLR
05.06.2015	Introduction to section 2.1.4 Human Operator Models	Simona Collina, SVN
22.06.2015	Added Task models and Human operator model description, some structural changes	Jan-Patrick Osterloh, OFF
23.06.2015	Review Version	Sara Sillaurren, TEC
26.03.2015	Internal review	Elisa Landini, REL
06.07.2015	Final document for delivery	Sara Sillaurren, TEC

## Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>6</b>
1.1	Overview on Model-based Design .....	6
<b>2</b>	<b>Common Modelling Framework.....</b>	<b>8</b>
2.1	Task Model .....	8
2.1.1	Introduction .....	8
2.1.2	General Application of Task Models.....	10
2.1.3	Application of task models in HoliDes.....	12
2.1.4	Types of task modelling.....	13
2.1.5	State-of-the-Art .....	14
2.1.6	HoliDes task modelling .....	16
2.2	Resource Modelling Language .....	26
2.3	Cooperation Models.....	29
2.3.1	Modelling the cooperative system.....	29
2.3.2	Modelling the agents cooperation .....	30
2.4	Human Operator Models .....	33
2.4.1	CASCaS .....	34
2.4.2	COSMODRIVE .....	36
2.4.3	CPM-GOMS.....	37
2.4.4	Pilot fatigue model.....	38
2.4.5	Cognitive driver distraction model .....	39
2.4.6	The MDP/MDPN Co-pilot model .....	44
2.4.7	The DBN driver model .....	49
2.5	HMI Interaction Models .....	56
2.5.1	The problem of HMI programming.....	56
2.5.2	A new event-based approach .....	60
2.6	Training Models .....	65
<b>3</b>	<b>Modelling Techniques and Tools V1.5 .....</b>	<b>67</b>
3.1	MagicPED (OFF).....	69
3.1.1	Introduction .....	69
3.1.2	State-of-the-Art .....	69
3.1.3	MTT Description .....	72
3.2	Human Efficiency Evaluator (OFF).....	76
3.2.1	Introduction .....	76
3.2.2	State-of-the-Art .....	78
3.2.3	MTT Description .....	80
3.3	TrainingManager (OFF, TRS).....	84
3.3.1	Introduction .....	84
3.3.2	State-of-the-Art .....	84
3.3.3	MTT Description .....	85
3.4	Djnn (ENA).....	91
3.4.1	Introduction .....	91

3.4.2	State-of-the-Art .....	91
3.4.3	MTT Description .....	92
3.5	Human Operator Models .....	96
3.5.1	CASCaS (OFF) .....	96
3.5.2	COSMODRIVE and COSMO-SIVIC (IFS) .....	99
3.5.3	GreatSPN for MDPN (UTO) .....	103
3.5.4	Bayesian Autonomous Driver Mixture-of-Behaviours Models - BAdMoB (OFF) .....	109
3.5.5	Driver Distraction Classifier (TWT).....	120
3.5.6	Pilot Pattern Classifier (TEC).....	123
3.5.7	Driver Distraction Classifier (UTO).....	126
<b>4</b>	<b>Integration Plans.....</b>	<b>136</b>
<b>5</b>	<b>Requirements Update .....</b>	<b>137</b>
	<b>Table 6: Overview of current requirements status .....</b>	<b>138</b>
5.1	Changes in the status of the deliverable for WP6.....	138
5.2	Changes in the status of the deliverable for WP7.....	139
5.3	Changes in the status of the deliverable for WP8.....	139
5.4	Changes in the status of the deliverable for WP9.....	140
<b>6</b>	<b>Summary .....</b>	<b>142</b>
<b>7</b>	<b>References.....</b>	<b>143</b>

## 1 Introduction

The development of an interactive system is a complex problem, which is continuously growing with the evolving technology, i.e. growing cooperation between actors in distributed locations, or future application of adaptive systems. Thus, a good Human Factors Design is essential to provide safety in these complex systems, especially concerning the human-machine interaction. Human Factors (or Ergonomics) is the scientific discipline concerned with the understanding of interactions among humans and other elements of a system, and the profession that applies theory, principles, data and methods to design a system in order to optimize human well-being and overall system performance [74]. In essence, it is the study of designing equipment and devices that fit the human body and its cognitive abilities. Within HoliDes, Human Factors is understood especially to build systems, which are adapting to the cognitive abilities of the human operator. *D1.5 and Annex II of D1.5 (HF Method Library)* lists human factors methods. For some of these methods, WP2 will provide formal modelling languages that support the modelling of adaptive and cooperative Systems (AdCoS), as well as editors for the specification of these models. The modelling languages can be used to model the AdCoS in WP6-9. In WP4 evaluation methods are developed based on the models defined in WP2. The models will also be used in WP3 to define and analyse Adaptation, and are employed to guide design and evaluation in WP5. The developed models will contribute to the Common Meta-Model of the HF-RTP in WP1.

Model-based approaches can be very helpful to manage system complexity, because models can be described on different levels of abstraction, focused on the relevant information in a structured way. One advantage of model-based approaches is that these models can be analysed in multiple ways, e.g. they can be checked for consistency, safety (formal methods) or efficiency.

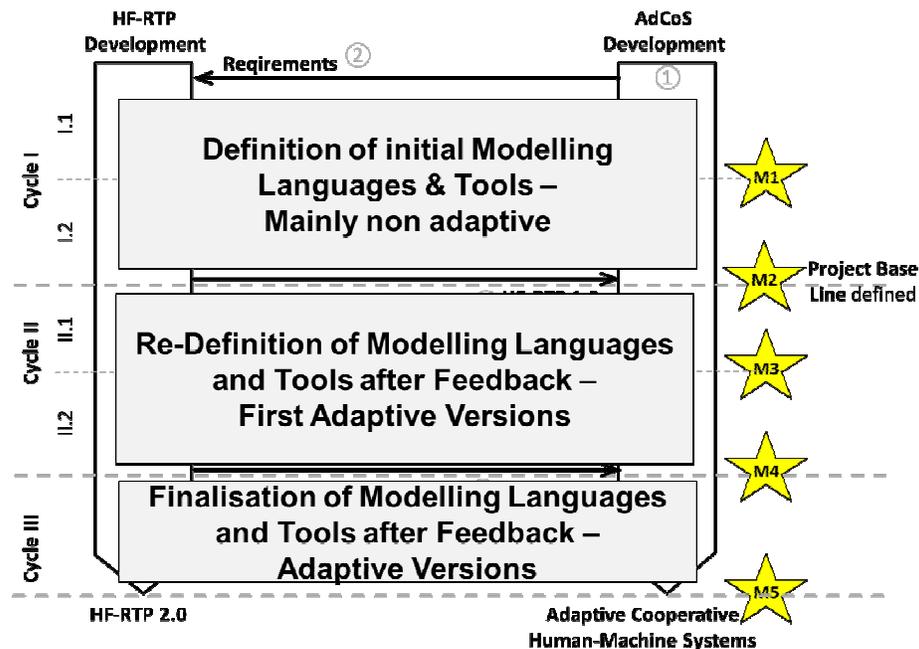
### 1.1 Overview on Model-based Design

Model-Based Design (MBD) is a method for addressing problems associated with designing complex systems, and is based on syntactically and semantically (e.g. mathematically) defined abstractions of the system and the environment and the interactions between them. MBD is widely used in e.g. aeronautics and space domain, but usage could be improved in all domains.

Model-based design allows developing complex systems, because the models allow easier communication and involvement of other experts, due to the graphical visualisation of the model, as well as the defined semantic of the model. Main benefit and cost saver is probably the code-generation facilities of the MBD. In addition, simulation of the model allows for easier testing and thus improvement of the product quality, while gaining shorter development at the same time. This is strengthened by the use of code generation, and support for

model-based analysis, i.e. verification and validation. Verification and validation are two major system engineering technical processes (ISO IEC 2008). Verification focuses on technical requirements coming from the engineering point of view (and not from the user point of view). Verification tries to answer the question "Are we building the system right?" Contrary to verification, validation deals with final user and operational related requirements, trying to answer to "Are we building the right system?" Model-based analysis is a major approach to support verification and validation processes. The idea is to construct an intermediate representation of the future system – the model – and to search for evidences directly on this representation. V&V is tackled in WP4, see *D4.4* for more details.

Nevertheless, in the current industrial practice, there is only poor support for Human Centred Design and associated Human Factor Analysis, especially for adaptive systems. In HoliDes the MBD for AdCoS incl. Human Factors will be tackled by defining or choosing appropriate existing modelling languages, allowing designers to model also adaptation as well as human behaviour and analysis. In Figure 1, the three cycles of HoliDes, in which the modelling languages are developed and evolved, is depicted.



**Figure 1: HoliDes Cycled Approach**

## 2 Common Modelling Framework

In WP2, modelling languages are developed, which allow formalizing different aspects of an AdCoS. These modelling languages are described in the following sections. Section 3 will then describe tools and methods that are developed within WP2, are based on the developed modelling languages, and allowing to instantiate the models in associated editors. In later versions of this deliverable, the described models will be interconnected in the common modelling framework.

### 2.1 Task Model

#### 2.1.1 Introduction

In general terms, task modelling is concerned with describing how the work is performed by one or more persons to achieve a given goal. However, such a loose definition also means that a primary concern in the planning of the model is the desired level of granularity, which can go from a general level down to specific hand movements on a control panel, for instance. A modelling language is intended to express the properties of the modelled entity in a way that covers all the aspects needed to fulfil the purpose of the development work. It is therefore useful to look at what problem domains a task modelling language is expected to cover and what the models needs to express to be helpful in that domain.

Task models are attempts at describing tasks, subdivisions of a sequence of activities, in such a way that they can be treated formally and provide a useful level of predictability for their intended purpose. As the intended purpose of task models in HoliDes varies across the application domains, a clarification of the matter is in order. The following sections will discuss the notion of a task and a task model before going into the task modelling employed in HoliDes.

The focus will be on systems and their environments that can be described in terms of a state. 'State' here refers to some set of values (whether discrete or continuous) that together provide a complete description of the system and environment, in such a way that the future state can be calculated for a given known input or disturbance.

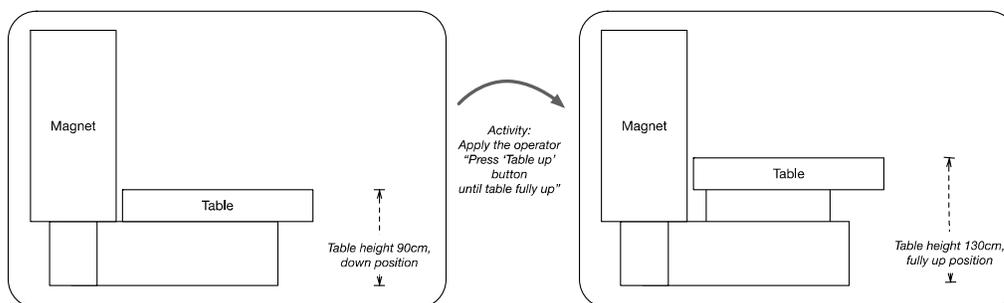
When designing an AdCoS – or any other system for that matter – a necessary prerequisite is to understand what the people using the AdCoS are expected to do, as well as how they achieve the goals that are set for them, be it by themselves or some other governing process or regulation. Task models are a method to capture this type of insight into people's work and to help improve systems whether at the design stage or applied to an existing and deployed product. For the purposes of AdCoS design work, where a sequence of activities

can often be seen as a series of state transitions of the controlled entity, a task can be characterised by three properties:

- 1) A goal state
- 2) A required specific initial state
- 3) An operator, to create the transition from initial state to goal state

The term “state” above refers mainly to the state of the controlled entity, for example a task aimed at changing the AdCoS by entering information on the interface. The act of applying the operator is called an *activity*.

As an example of a task described in this framework, consider a simple task taken from the guided patient positioning use case of the healthcare domain. The task – and activity - is described in an informal language as “Hold 'Table-up' button until table is fully up”, as shown in Figure 2.

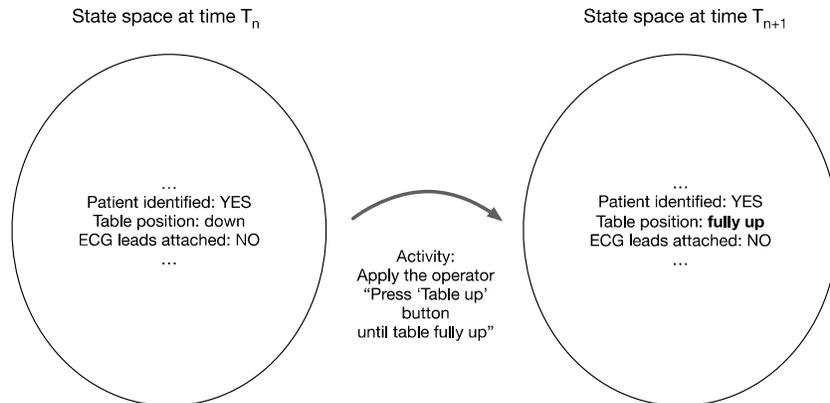


**Figure 2: Table being moved from down position to fully up by performing the activity defined in the example task.**

The table position is a precisely measurable value, expressing a specific state of the system that also determines if certain tasks can be performed. It is therefore a good state variable, and task descriptions, especially initial state and goal state can be based on it.

In more technical terms, the activity of pressing the “Table up” button produces a transition from one state to another. This is illustrated in Figure 3. Notice that only the state variable referenced in the task description (table position) changes, while the other state variables remain unchanged.

Adopting a mathematical term from control theory, the theoretical combination of all possible values of the state variables is said to make up the *state space* of the system being modelled.



**Figure 3: A state space defined by state variables depicted at two moments, before and after a state transition that causes a single variable to change.**

As can be seen, the initial state of the task is a state that the environment must be in order for the operation to be possible or meaningful. In the example above, there is no need to carry out the activity raising the table if it is already in the fully up position.

This means that the initial state of the task (which in reality is a requirement on the initial state of the environment) becomes a *condition* for the task to be executable. If the environment is not in the prescribed initial state, the task cannot be carried out. In computer science, such a condition is called a precondition, and expresses conditions that must be valid before a given operation can be invoked. Similarly, the goal state expresses a state of the environment that will be reached upon successful completion of the activity in the task. In computer science, the expression of the goal state of the task is known as a post-condition, in that it is a condition that expresses whether the operation has been correctly performed.

This leads to a possible simple, but more formal, definition of the task:

Task name	Move table to fully up position
Goal state (postcondition)	Table in fully up position
Initial state (precondition)	Table in a down position
Operator	Hold 'Table-up' button until goal state is met

For the purpose of such a simple example "down" position means any position below the "fully up" position.

### 2.1.2 General Application of Task Models

Within each large application domain, a space of issues and properties can be identified. These will provide a good reference to the properties of tasks,

resources and environment that need to be included for the model to be useful in the intended application domain.

The work of Stanton [156] can be used to identify the elements of tasks and resources that should be covered by the modelling language. The paper, which is based on previous work by Piso [140], Hodgkinson & Crawshaw [64] and Bruseberg & Shepherd [26], lists a series of questions used to elicit the knowledge about the domain itself and how work is done (or should be) in it. Stanton [156] covers the following application areas:

- Training Design
- Interface Design
- Job Design

For each area, the authors have identified the questions listed in Table 1, Table 2 and Table 3, and in the cases where the questions pointed at improvements or were intended to identify unacceptable workload, they have been changed into a more neutral wording for the purpose of the modelling process. In the tables, this has been recorded in the Notes column.

These knowledge elicitation questions can be used in the modelling phase of the AdCoS development process to help the modeller cover the relevant parts of the domain and the activities needed by the operator to perform the required work.

<b>Training design</b>		
<b><i>Knowledge elicitation question</i></b>	<b><i>Category</i></b>	<b><i>Notes</i></b>
What is the goal of the task?	• Goal-means hierarchy	
What information is used for the decision to act?	• Information flow • Decision making	
When and under what conditions does the person (or system) decide to take action?	• Decision making	
What is the sequence of operations that are carried out?	• Task sequencing	
What are the consequences of action and what feedback is provided?	• State space • Information flow	
How often are tasks carried out?	• Cognitive load • Workload	
Who carries the tasks out?	• Task allocation	
What kinds of problems can occur?	• Error handling	

**Table 1 Knowledge elicitation questions for task modelling intended for training design. Originally by Piso, 1981, adapted from Stanton, 2006**

<b>Interface Design</b>		
<b>Knowledge elicitation question</b>	<b>Category</b>	<b>Notes</b>
What are the sensory inputs?	<ul style="list-style-type: none"> <li>Information flow</li> </ul>	
What information is displayed on the UI	<ul style="list-style-type: none"> <li>Information flow</li> </ul>	Adapted
What are the information processing demands?	<ul style="list-style-type: none"> <li>Information flow</li> <li>Cognitive load</li> </ul>	
What kind of responses is required?	<ul style="list-style-type: none"> <li>Information flow</li> <li>Decision making</li> </ul>	
Which control inputs are provided?	<ul style="list-style-type: none"> <li>Task operator</li> <li>Information flow</li> </ul>	Adapted
What kind of feedback is given?	<ul style="list-style-type: none"> <li>Information flow</li> </ul>	
How do the control inputs relate to the goals?	<ul style="list-style-type: none"> <li>Goal-means hierarchy</li> </ul>	Adapted
Which environmental disturbances are present	<ul style="list-style-type: none"> <li>Error handling</li> </ul>	Adapted

**Table 2: Knowledge elicitation questions for task modelling intended for interface design. Originally by Hodgkinson & Crawshaw, 1985, adapted from Stanton, 2006. The questions marked "adapted" have been modified with respect to the original versions for use in task modelling.**

<b>Job Design</b>		
<b>Knowledge elicitation question</b>	<b>Category</b>	<b>Notes</b>
How does information flow in the task?	<ul style="list-style-type: none"> <li>Information flow</li> </ul>	
When must tasks be done?	<ul style="list-style-type: none"> <li>Task planning</li> </ul>	
What is the temporal relation of tasks?	<ul style="list-style-type: none"> <li>Task sequencing</li> </ul>	
What are the physical constraints on tasks?	<ul style="list-style-type: none"> <li>Outside constraints</li> </ul>	
Where can and cannot error and delay be tolerated?	<ul style="list-style-type: none"> <li>Error handling</li> </ul>	
What is the cognitive workload	<ul style="list-style-type: none"> <li>Cognitive load</li> </ul>	Adapted
What are the knowledge requirements for this task?	<ul style="list-style-type: none"> <li>Skills-Rules-Knowledge requirements</li> </ul>	Adapted
What are the skills requirements for this task?	<ul style="list-style-type: none"> <li>Skills-Rules-Knowledge requirements</li> </ul>	Adapted

**Table 3: Knowledge elicitation questions for task modelling intended for job design. Bruseberg & Shepherd, 1997, from Stanton, 2006.**

### 2.1.3 Application of task models in HoliDes

Annex II shows some examples from the AdCoS work packages on how task analysis has been used.

## 2.1.4 Types of task modelling

As can be seen from the knowledge elicitation questions listed in Table 1, Table 2 and Table 3, different purposes of the model lead to different types of domain knowledge being sought, and ultimately will lead to different models.

On a general level, three different main categories of task analysis and modelling have been identified (see for instance [65]) – descriptive, normative and formative models. A brief explanation follows:

### 2.1.4.1 Descriptive

Descriptive task models capture knowledge about how a system is operated, whether that is the ideal way of doing it or not. They are for natural reasons mostly applied to existing systems, but can also be used on simulated interfaces or through other mock-up techniques.

Descriptive task modelling is useful for providing critique of an existing design or organisation of specific processes, for instance with the purpose of identifying weak spots that need more work in the design process or to document existing procedures.

### 2.1.4.2 Normative

Normative models contain knowledge about how a work process, which can be designed around a technical system, should be organised.

They often use abstractions as a means to encapsulate higher-level knowledge, and a goals-means structuring of the model is a popular way to do this, as goals are a natural way to express desired outcomes without resorting to detailed specifications of behaviour. Hierarchical task analysis (HTA, see section 2.1.5.1 below) and Goals, Operators, Methods, and Selection rules (GOMS, see section 2.1.5.2 below) are two used approaches to build task models in the normative category, although HTA also often includes descriptive elements, especially at the lower levels of abstraction.

The normative approach to task modelling is useful as an element in a design process where the system is being designed from scratch, and the only available knowledge about the system is from the designers themselves – as there are no users yet to explain how the system works in reality.

### 2.1.4.3 Formative

The formative approach to task modelling tries to capture what can be done with the system [78]. The functions identified this way which can be more extensive than what the system was intended for by the original design.

The three types of models can be summarised as in Table 4.



Type of model	Task model		
	Descriptive	Normative	Formative
What the model expresses	How things are	How things should be	What things are possible
Example of modelling technique	Link analysis	Hierarchical task analysis	Cognitive work analysis

**Table 4: Overview of task model categories. Adapted from [65], p208**

### 2.1.5 State-of-the-Art

The HoliDes task modelling language combines three frameworks into one coherent system.

This covers task modelling from high-level concrete goals to more abstract lower-level goals. A formal representation that is suitable for implementation in software is often used.

The three elements on which the modelling language is based are listed below:

- The Hierarchical Task Analysis (HTA) modelling method, which focuses on a recursive breakdown of activities into a hierarchy of goals, plans and operators. HTA does not provide a modelling language in itself, but provides the elements that the modelling language must be able to describe.
- GOMS (Goals, Operators, Methods, and Selection rules), which is a modelling approach often used to analyse low-level tasks and actions, for instance down to keystrokes on a keyboard.
- The W3C task model framework, based on the CTTE notion is a meta-model describing the structure of actual task models.

Brief descriptions follow:

#### 2.1.5.1 HTA

HTA – hierarchical task analysis – is widely used in variety of domains and contexts (incl. interface design) [9]. The main characteristics of HTA can be summarized as follows:

1. Work is decomposed into a hierarchical structure:
  - Goals and sub goals – what the user wants to achieve
  - Tasks – what the user must do to achieve these goals
  - Subtasks – smaller and lower level steps that make up the tasks
2. Plan analysis
  - Order in which the activities are to be carried out
3. Structured output

- Hierarchical task description e.g. tree or tabular diagram

The subtask breakdown should be repeated until the desired level of granularity has been reached. The plan analysis can typically be expressed by sequencing rules that express if the subtasks must (or can) be carried out in any specific order. Examples of sequencing rules are:

- Linear
- Simultaneous
- Cyclical
- Branching

The approach based on a decomposition of tasks means that the analysis method has a strong top-down bias. This means that the top-level goals tend to be abstract and normative, while the lowest levels of the model will be mainly descriptive [78].

### 2.1.5.2 GOMS

The GOMS model was developed by Card, Moran and Newell [28] as a way of quantitatively predicting the skilled and error free performance of users interacting with a text editor, and is now commonly referred as CMN-GOMS:

"For error-free behaviour, a GOMS model provides a complete dynamic description of behaviour, measured at the level of goals, methods, and operators. Given a specific task (a specific instruction on a specific manuscript and a specific editor), this description can be instantiated into a sequence of operations (operator occurrences). By associating times with each operator, such a model will make total time predictions. If these times are given as distributions, it will make statistical predictions" ([28], p.146).

Since then, GOMS has been widely extended for use with other categories of HMIs (e.g. KLM, NGOMSL, CPM-GOMS, etc.).

GOMS takes its name from the main elements that make up a task model created under this scheme:

Goals:	Task decomposed into nested hierarchy of goals and sub-goals
Operators:	Hierarchy ends in operators, whose actions cause transitions between states
Methods:	Sequences of operators executed to accomplish a set of sub-goals
Selection rules:	Rules that determine which method to use

A GOMS model consists of *goals* that can be achieved by applying specific *methods*, which at the lowest level are composed of *operators*. The operators are specific steps that a user performs and are assigned a specific execution time. Whenever a given goal can be achieved through more than one method,

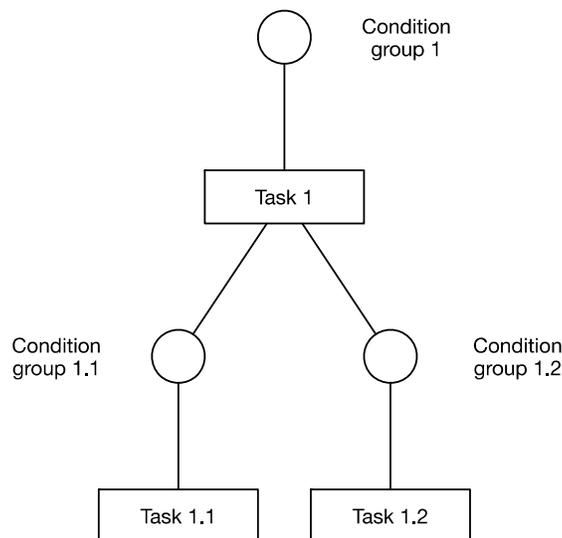
*selection rules* are used to determine the proper method. GOMS models are often used to model low-level tasks, for instance the use of a keyboard.

### 2.1.5.3 W3C

The W3C work on task models provides a formal framework for task modelling that spans from goals to activities. Task models expressed in this format describe the tasks that must be performed to achieve the stated goals [168], and the aspects of the world that can be covered by the models are defined by the meta-model. The meta-model contains several classes, but the ones most central to this discussion are *Task* and *Condition Group*.

A simple example is provided in Figure 4, where a high level goal (expressed in a condition group, *Condition group 1*) can be achieved through a single task (*Task 1*). For the purpose of this example, *Task 1* has two preconditions that need to be fulfilled, namely *Condition group 1.1* and *Condition group 1.2*. Each of the condition groups 1.1 and 1.2 contain goals that can be achieved by *Task 1.1* and *Task 1.2*, respectively.

For simplicity, the actual conditions in the condition groups are not shown, but a more detailed graphical representation would depict them as well.



**Figure 4: A simple example of tasks and condition groups.**

As stated above, the description here is deliberately simple. For details of the meta-model, please refer to [168].

### 2.1.6 HoliDes task modelling

After the initial task modelling language has been proposed (see D2.4) partners have provided their own task modelling languages, which are used by the

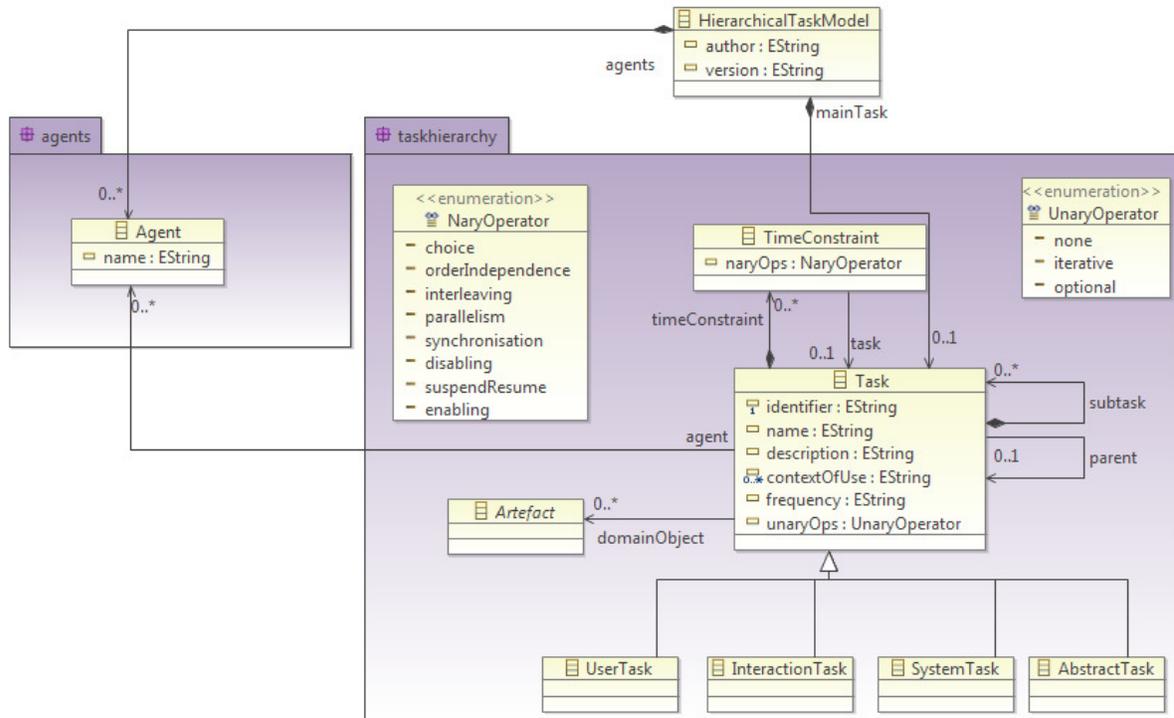
partners (before and within HoliDes). Discussion about the HoliDes task modelling language has been started, based on the contributions of the partners. For that reason, the previously called HoliDes task modelling language (D2.4) will be referenced now as the OFF task modelling language. The following sections will provide an overview on the currently collected task modelling languages from OFF, AWI, HFC and PHI. Three more partners will contribute their languages in the next month. As soon as this collection is completed, the consortium will discuss these models and integrate it into (one or more) common task modelling languages. Probably there will be more than one common task modelling language, as the underlying task analysis methods may have very different concepts, which are not expressible in one common model.

#### **2.1.6.1 OFF task modelling language**

The OFF task modelling consists of two parts. The first one is based on hierarchical task modelling (HTA), especially regarding the task hierarchy. Planning (which is also a part of many HTA schemes) is supported through the use of "time constraints" between the tasks: *before*, *parallel*, *choice* (or without any constraints it is unordered). This provides the HoliDes task model language with a high level of modelling capability, which typically will encapsulate normative task knowledge. The formal structure of the models is based on the work of W3C.

The second parts extends the modelling with a lower level, which will provide a more descriptive modelling of the actions close to the actual physical equipment, software UI, other agents and the controlled entity. This level of the modelling is based on GSM (Goal-State-Means) modelling, to form an overall modelling framework that connects the higher-level goals of the top level model with the actual state of the environment.

The task hierarchy package is part of the HF-RTP Meta-Model, and described in more detail in deliverable D1.4. Figure 5 shows the task hierarchy as UML Diagram.



**Figure 5: OFF Task Hierarchy Model**

Figure 6 shows the GSM based modelling level as a UML diagram.

Each task is associated with a set of rules, i.e. each task can have one or more rules associated. These rules allow a very detailed level of task modelling. Adding rules to the tasks is optional and only necessary when a) the tasks should be used within CASCaS for simulation of human behaviour, or b) one of the following analyses provided in future versions should be performed:

- Execution time
- Workload
- Task/Procedure Comparison

Figure 6 shows the UML model for the Rule layer. Each rule consists of a left hand side (LHS) and a right hand side (RHS). The LHS forms the IF part of a rule, and the RHS the THEN part of a rule. LHS elements consist of Memory Read items, to retrieve memory variables, and Conditions on the variables. RHS elements are actions executed when the rule itself is executed. These are Memory Store, Motor, and Voice actions. The first assigns new values to memory variables, the latter two enable direct manipulation of environment variables.

There are three types of rules, regular rules, percept rules, and waiting rules:

- A regular rule is fired if its task is the active task and if the conditions in its LHS evaluate to true. The conditions are made on Memory Variables, which often serve as internal representation of Environment Variables. Thus, memory retrievals are necessary to check the condition on the rule.
- Percept rules are also triggered by their task, but only if an environment variable is not encoded in a memory variable, which is needed for one of the regular rules of the task.
- Waiting rules have no LHS and RHS elements, and are fired when neither a regular nor a percept rule can fire.

For achieving a task, it is necessary that a regular rule for that goal is fired and all the sub-tasks are achieved. Firing a percept rule does not make the task finished, i.e. the task stays active. Firing a waiting rule allows interleaving with other tasks, i.e. if there are other tasks that can be selected (not yet achieved and not active, i.e. in case of interleaved time constraint), these tasks can become active, and the old active task will be reactivated later again. In the following, the elements of the rule in the LHS and RHS explained in more detail.

#### 2.1.6.1.1 LHS: Items of the condition pattern

The left hand side of a rule can be considered as a *search pattern* across the models memory. If the pattern that consists of **Retrieve** and **Condition** statements can be matched, the rule is selectable.

##### **Retrieve(variable, age)**

The retrieval request searches the memory for the occurrence of the specified *variable*. A variable is a chain of associative links that point to a specific node, typically an object or an attribute of an object. The *age* parameter specifies that the node specified by the **Retrieve** command must not be older than the given *age* value. *Older* means that the last time this node was written by the perception or by an explicit *Memorize* statement was not before current simulation minus age in milliseconds. This has nothing to do with remembering and forgetting, it is additional knowledge that certain information has to be "up-to-date".

If a task contains regular rules with retrieval statements that require time-critical information, one has to consider that those rules are not selectable if the information is outdated. If no regular rule is selectable a waiting rule is chosen instead (if specified) but in most cases at least one additional regular rule should complement the rule set for this task which contains a **LookAt** command. This Command moves the gaze towards the object and updates the required information in the memory component.

##### **Condition(boolean expression)**

The condition statement checks a boolean expression. If the condition cannot be evaluated to "true" the rule will not be selectable. The boolean expression can

contain any number of checks across a number of memory path elements. The possible operators that can be used within the expressions are:

- &&, ||
- <, >, <=, >=, ==, !=,
- +, -, \*, /,
- (, )

The elements that can be compared are

- memory paths (which refer to concrete values). Nodes without a value cannot be compared.
- numbers (floating point and integer)
- String variables can contain any number of the following chars: *a-z, A-Z, 0-9, \_, -* and they are embraced by single quotes ` ` , e.g.: *'hello\_123'*, *'yes'*  
...

### ***RHS: Items of the action pattern***

If the left hand side was matched successfully the rule is fired, the right hand side (action pattern) is executed. The action pattern can contain a number of commands that trigger motoric actions, move the visual perception, to actively memorize certain values or to add certain relations (associative links) between nodes.

### **LookAt(memory path)**

The model shifts its visual attention towards a certain object using a coordinated head / eye movement. The object is specified through the parameter **memory path** and is either a variable, an AOI or an object type matched in the LHS.

### **Motor(resource, type, memory path, value, guidance)**

The model initiates interaction with the environment, e.g. pressing a knob, dialling in some values or using a steering wheel and the pedals of a car. A motor command always triggers a sensory-motor pattern on the autonomous layer, which may run in parallel to the associative layer. Interference with other tasks may occur if the required input is not available because the model's visual attention is not targeted towards the necessary information sources. Attentional distraction due to task interleaving is currently not part of the model.

1. **resource** can be either
  - left or right hand, left or right foot
2. **type** can be:
  - Move: move the hand to the location of an object
  - Unguided\_move: unguided move to the resource (without the visual feedback of the eyes).
  - Grasp: grasp the object, if resource not already moved to this instrument, move action is automatically done
  - Release: release the object (precondition: grasped object before)
  - Adjust: adjust the object to a new value (e.g. dial in a value in a potentiometer, shifting the gear, steer the wheel, ...)

- Type: type in a word or value into a keyboard (like AOI (a grasp of the keyboard has to be executed before))
  - Mouse-move: move the mouse to a new location (a grasp on the mouse has to be executed before)
  - Mouse-Click: click with a mouse on a location (precondition: mouse grasped)
  - Mouse-double-click: click with a mouse twice on a location (precondition: mouse grasped)
  - Push: push a physical button
  - Pull: a physical object (e.g. altitude selector, direction indicator)
3. **variable** can be any resource (i.e. variable, object). For objects of those type at least one output channel must be defined, otherwise the Motor command will throw an exception.
  4. **value** depends on the defined type and the memory path element that is referred to. If the memory path points to an integer data type, e.g. *lever.position*, value must contain an integer number.

#### **Memorize(memory path, value)**

The model can explicitly memorize additional information which it concludes from the current situation. These conclusions may be remembered (**Retrieve**) within the search pattern of any other rule. If the parameter *memory path*:

1. Does not exist, a new path is added as specified and the value is appended as destination node of the path.
2. Was partially matched in the search pattern, all path elements that were not matched are created and the value is appended as destination node of the path.
3. Is fully matched, a new value is appended to the path.

#### **TaskDone(task name)**

This statement can be used to terminate a task immediately. If this item is fired, the task module searches for the existence of this task and the task is removed. This statement results in a recursive descent through all subtasks which are also removed from the module. Important: This statement is the only possibility to terminate iterative tasks.

#### **Workload Annotation**

Each action element in the rules allows adding workload annotations, according to the workload theory of McCracken & Aldrich [125].



# HoliDes

Holistic Human Factors **Design** of Adaptive Cooperative Human-Machine Systems

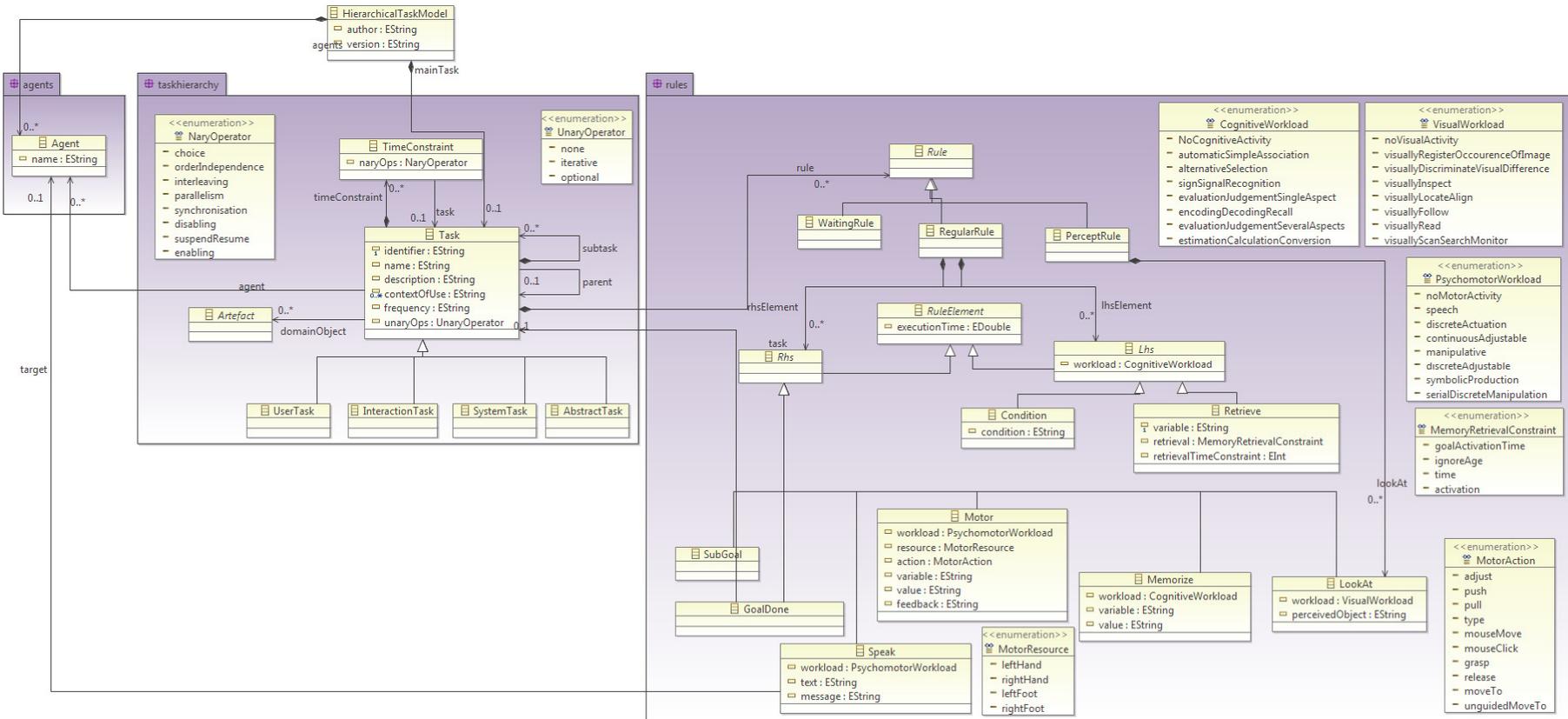


Figure 6: Rule Level Model



**2.1.6.2 AWI task modelling language**

Means-end modelling, hierarchical decomposition								
Goal	Relation	Subgoal	Relation	Objective	Relation	Function	Relation	Behaviour
<b>Goal</b> Description [ID]	[Decomposition]	<b>[Subgoal]</b> Description [ID]	Specify	<b>Objective</b> Description [ID] Quantifiable property	Achieve Optional?	<b>Function</b> Description [ID]	Provide Optional?	<b>Behaviour</b> Description [ID] Formal action
							Provide-choice	<b>Behaviour</b> Description [ID] Formal action

**Figure 7: AWI Task Modelling Language**

The means-end model consists of a hierarchy of elements that span from goals (abstract) to behaviour (actions, concrete). These elements are connected through relations, which most often are just links between an element (e.g., a function) and the lower-level elements that support it (e.g., behaviour).

The relations are mostly just connections, but in some cases they can be tagged as "optional", or they can be a relation-choice (e.g., Provide-choice) in cases where there is more than one way to provide the function or achieve an objective, for instance.

	<p><b>HoliDes</b></p> <p><b>H</b>olistic Human Factors <b>D</b>esign of Adaptive Cooperative Human- Machine Systems</p>	
--	---	---

All elements have an optional ID, which can be used to refer to the element in other parts of the model. This is needed in cases where a single function is used to achieve more than one objective for instance. In that case, the function is defined in a relation to one objective, and simply referred to by its ID in any subsequent references.

### 2.1.6.3 HFC task modelling language

HFC is currently developing a procedure to support task analyses (i.e. data gathering towards a task model), and uses a Hierarchical Task Analysis method for the first steps. Figure 8 shows the current version of the modelling language (under development). Up to now, there is a hierarchy of (sub-) tasks with their relations and features defined, as well as a general rule as of how to “walk through” the entire task. Goals are not defined explicitly, but are implicitly reached when the task is performed successfully.

Each task has an “ID” number and a short textual “*description*”, which should be solution independent.

In contrast, “*required action*” is the solution dependent counterpart. For instance, a description could be “Choose protocol”, whereas the required action would be “Press protocol button”. This separation is comparable to a differentiation between goals and concrete tasks.

The “*controls / displays used*” field should be filled in with the system parts the user is interacting with, e.g. “Touchscreen 2”.

To define hierarchical relations, the ID of the parent task is also saved for each task. The “*type of task*” is either “mandatory” or “optional”; “*branching node*” can have the values “recall”, “decision”, “check” or “none”.

“*Type of task*” together with “*branching node*” and the general rule of the model define the user’s path through the model; i.e. how the task procedure is performed.

“Modality” means that a task can optionally be tagged “speech”, “manual”, “auditory”, “visual”, “cognitive”, or “system”, depending on whether the task requires the user to use sensory channels or do something with their own body. Additionally, the tag “system” means that there is a system reaction needed by the user in order to proceed. In “Description of system feedback / reaction” it is described what this entails.

Up to now, there are no actors or user roles defined, but this is planned for the future.

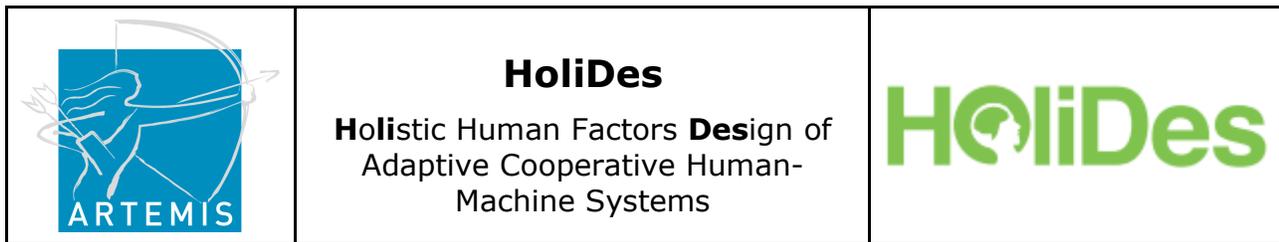
HFC-HTA_Task-Model				
Model	Class	Attribute / Rule content	Type	Comment
HFC_HTA_TaskModel				
	Task			
		ID (Task number)	number	this entails information about successor / predecessor
		name / description	string	preferably solution independent
		parent task	number	
		type of task	{mandatory, optional}	
		branching node	{recall, decision, check, none}	
		controls / displays used	string	solution dependent. system-specific set of controls and displays is to be added before task analysis
		required action	string	solution dependent
		modality	speech, manual, auditory, visual, cognitive, system	tags (zero to all of these)
		description of system feedback / reaction	string	
	Rule			
		condition	string	Defines how to "walk through" the tasks. In case of the task being a branching node (decision/ check/ recall): If answer is yes/positive, proceed with next mandatory task. Otherwise, continue with the subsequent optional task(s), until the next mandatory task is reached.

**Figure 8: HFC Task Modelling Language**

#### 2.1.6.4 PHI task modelling language

PHI also uses a kind of hierarchical task analysis for their system design. The method is based on the work of Larry Constantine [32].





shows the ecore<sup>2</sup> Model of the 1<sup>st</sup>, yet unconsolidated, version of the HoliDes resource model.

The main class of the resource model is the abstract **Artefact** class, representing an arbitrary resource. By sub-classing this class, further refinements can be made:

- The **SoftwareArtefact** class represents any resource that is software. Currently there is only a sub-class for User Interfaces (class **UI**), which has to be substituted in future versions with the HMI Interaction model from WP2.5, see section 2.5.
- The **HardwareArtefact** class represents any resource that is hardware. This has been taken from DCoS-XML. Hardware is currently distinguished as **DiscreteActuator** (e.g. an on-off button, gear-shift), **ContinuousActuator** (e.g. the altitude selector of an aircraft's autopilot), **Consumable** or **Sensors**.
- The **EnvironmentalArtefact** class represents any resource in the environment, e.g. a **Space**. This is currently not further defined.

The Artefact is hierarchical, i.e. a Resource can have children. This allows building e.g. a complete cockpit, which consists again of many sub-resources. While the Artefact class and its subclasses describe the functional behaviour of the resource, one can associate a (not yet described) Shape with an artefact. These shapes will describe in future versions the visual parameters of a resource, primarily location, size, colours and form. In order to allow simulation, each artefact is also described by a set of attributes which describe the current state of the resource. Typically each artefact is represented as an Object (**Resource** class), which has **Attributes** of a certain **DataType**. The **MODE** shows, if the attribute is published or consumed by the resource.

---

<sup>2</sup> Eclipse modeling framework (EMF): <http://eclipse.org/emf>



# HoliDes

Holistic Human Factors **Design** of Adaptive Cooperative Human-Machine Systems

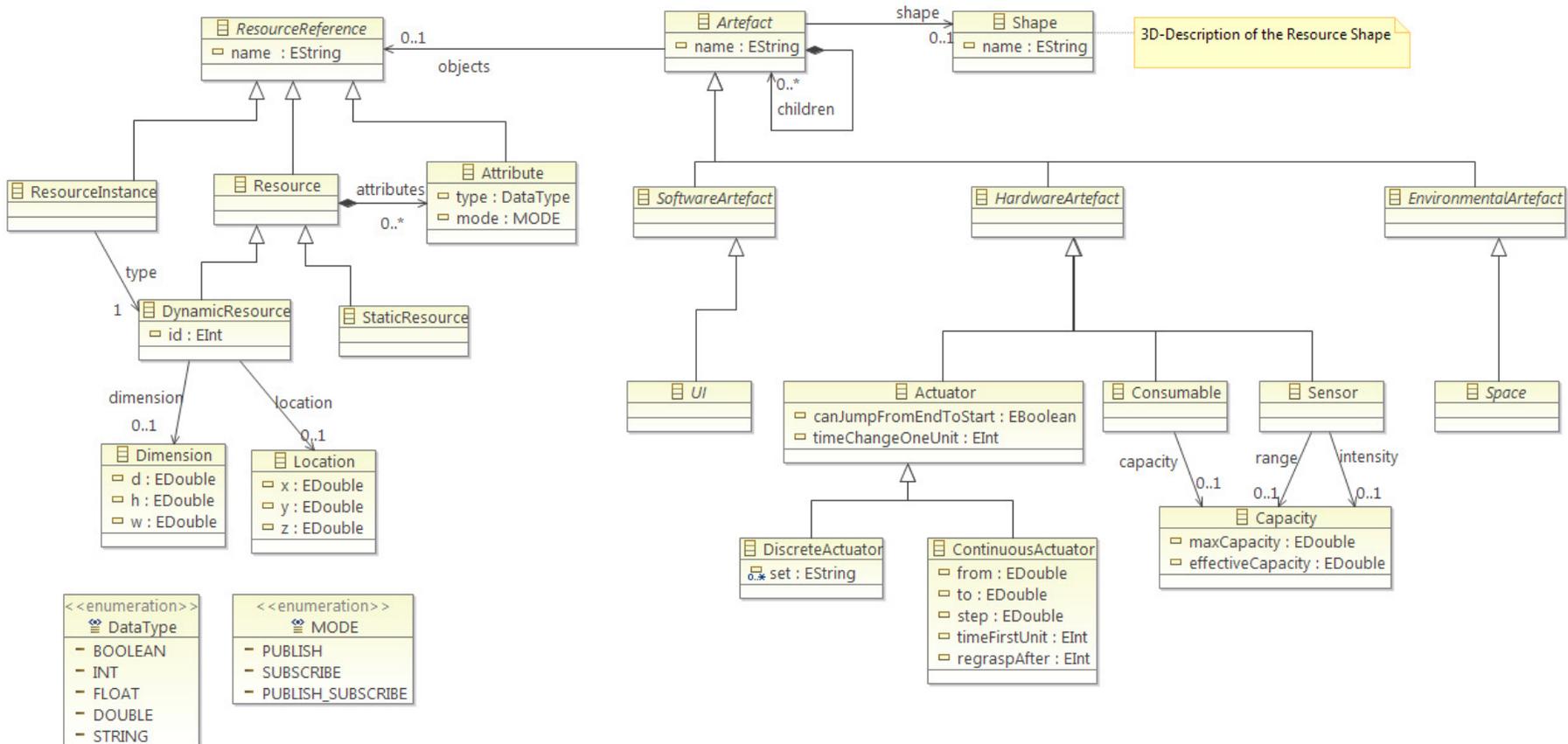


Figure 10: Resource Model

As this model is currently in beta status, and not yet discussed with the partners, further explanation will be skipped, until a consolidated version exists.

## 2.3 Cooperation Models

As presented in D2.4, in order to address the cooperation modelling, we started from the study of the approach used in the Artemis JU project D3CoS (Designing Dynamic Distributed Cooperative Human-Machine Systems [181]). Indeed, one of the aims of the D3CoS project was to model cooperation in multi-agent human-machine systems (briefly named “DCoS” – Distributed Cooperative human-machine Systems). The HoliDes system model can be considered as an adaptive specialization of the more general D3CoS system model (“AdCoS” – Adaptive Cooperative human-machine systems) where great attention is paid to human-factors aspects.

From the modelling perspective, the D3CoS project delivered:

- A modelling language for cooperative distributed human-machine system (DCoS-XML [36])
- A reference framework for analysing the cooperation between the agents of the system (D3CoS general framework [63])

Both modelling tools are conceived to enable the description of some aspects of the considered system in a formal way.

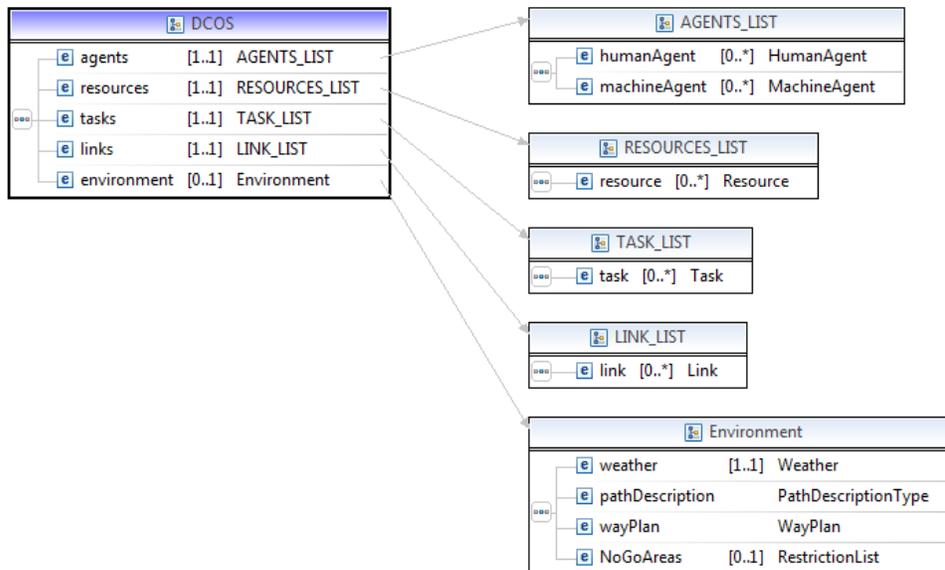
In particular, the first one focuses on the structure of the systems in terms of its main components (agents, tasks, resources, and environment) and related characteristics, while the second one on the types of cooperation that appear in the system dynamics.

### 2.3.1 Modelling the cooperative system

The DCoS-XML language [165,169] is an XML-based language for the description of DCoS defined by means of the standard XML Schema language. It specifies a data model capturing the main features of agents, resources and tasks envisioned in the system, as well as of the relations between such main components (links) and of the surrounding environment.

For each component, the DCoS-XML specifies a dedicated data structure including the attributes that have been considered relevant for the component characterization. For example, the environment data structure includes a set of weather information (visibility, wind condition, precipitation, date and time, cloud condition) and other features. Agent, tasks and resources are represented by means of specific data structures as well.

A model written in the DCoS-XML language is an XML file containing the XML descriptions of all the components of the systems according to the data structure that are associated to them (i.e., there will be a list of agents description, a list of resources descriptions, and so on). Figure 11 represents that concept.



**Figure 11: Top level elements of the DCoS-XML model**

The XML file corresponding to the model of the system (i.e., the DCoS model of the system) can be used as a common and non-ambiguous description of the overall system. As already pointed out in D2.4, besides the other advantages, the XML format is suitable for tool processing (like parsing, transformation towards UML diagrams, etc.) and for data exchanging between tools.

Within the WP2 context, the extension and specialization of the DCoS-XML model in order to obtain a formalism suitable to be applied to the HoliDes case have been considered and evaluated.

According to the agreed solution, in the HoliDes project, modelling languages for the formal representation of tasks, resources, and for human agent features will keep on being developed separately. This allows for a more focused and specialized modelling specification effort. On the other hand, the **HoliDes Common Meta Model** will play the inclusive role played in D3CoS by the DCoS-XML formalism. Indeed, the purpose of such a Common Meta Model is describing how to leverage and exchange all the developed models within the HF-RTP under development.

### 2.3.2 Modelling the agents cooperation

As motivated before, we consider as a starting point the D3CoS general framework for modelling cooperation between system's agents [184]. Such a framework is based on the so-called Hoc's framework [63].

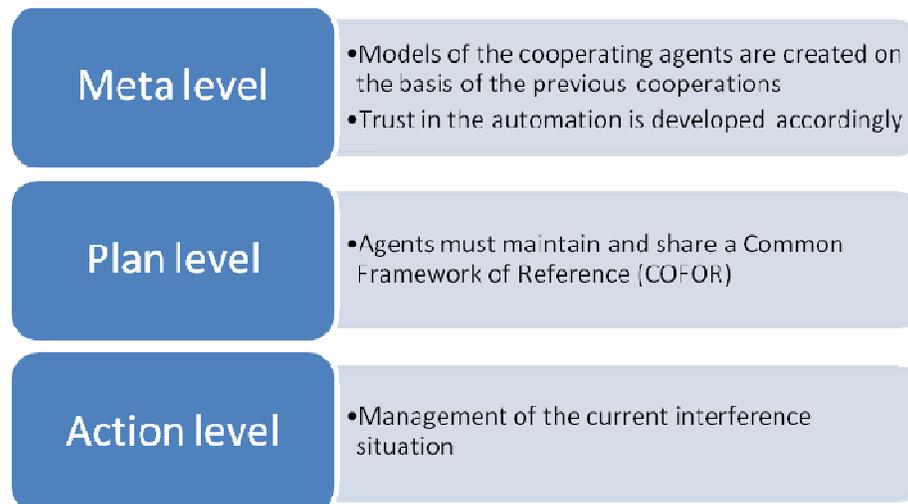
Hoc considers the relations between human and automation in *Human Machine Systems (HMS)* from the cooperation perspective. In particular, cooperation is considered in the so-called *dynamic situations*, i.e., when the behaviour of the

overall HMS cannot be fully determined by considering the actions of the HMS itself, since other factors, hidden and unexpected, come into play. This kind of situation is also characterized by high temporal constraints and risk, as typically appears in industrial process control, air traffic control, highly automated aircraft piloting, etc.

According to the Hoc's framework [63], collaboration or cooperation can be defined as follows: "*Cooperation is an activity of interference management between non-independent tasks distributed among several agents*", where the interference is intended to be managed in order to facilitate the individual tasks or the collective task if it exists.

The purpose of the Hoc's framework is going beyond the Sheridan's approach of Levels of Automation focused on the function allocation problem [185] from a machine-centred perspective in order to move to a more human-centred one. This new perspective is aimed to allow the classification of the interaction processes between the human and the machine from a cognitive point of view.

The framework envisions three incremental **Levels of Cooperation (LoC)**: (i) *action level*, (ii) *plan level*, (iii) *meta level*. These levels are identified in relation to the complexity of the interference management activity.



**Figure 12. Hoc's framework – Levels of Cooperation**

At the **action level**, the interference is managed for the time that it occurs. Managing interference requires an increased effort to the agents, but, on the other hand, it represents a means for adapting to unforeseen situation: some complexity cost must be paid to gain adaptation power.

At the **plan level**, there is the interest of sharing a **Common Frame of Reference (COFOR)** between the agents according to which manage the current and future interferences. The COFOR represents the information about

the environment and the interfering agents' activities: this information is exploited to facilitate the cooperation at the action level. Each agent must maintain its representation of the COFOR. Communication between the agents of the system is a crucial issue for maintaining a shared and effective COFOR, i.e., for being in a true "situation awareness" condition. The human must communicate their intent to the machine by means of his actions and the machine must communicate its activity to the human by means of suitable communication channels. Working on this communication is essential because the situation awareness is the precondition for optimizing the interaction between agents at the action level [61]. Moreover, poor situation awareness can be the major cause of difficulty to return to manual control in emergency situation [61]. The benefit of having a situation awareness condition (i.e., the sharing of an effective COFOR) is paid in terms of the effort in building and maintaining the COFOR among agents.

At the **meta-level**, an even bigger effort is paid in dealing with cooperation, since there is the interest of building and exploiting, besides the COFOR, also the experience of the previous cooperation within the team to optimize the interactions. Models of the behaviour of the other agents are elaborated by experience and the trust in the automation part of the systems is calibrated accordingly [102].

Besides the LoC, three different cooperative situations between the human and the automation parts of the HMS are identified. They are called **Modes of Cooperation (MoC)**: (i) *perceptive mode*, (ii) *mutual control mode*, (iii) *function delegation mode*.

In the **perceptive mode**, the machine is used as an extension of the sensorial organs, producing measures that must be passed through a suitable HMI to the human for having the human elaborate them.

In the **mutual control mode**, the machine controls the activity of the human in order to check if it is acceptable in terms of associated risks or with respect to some constraints. This kind of cooperation can be declined in further modes:

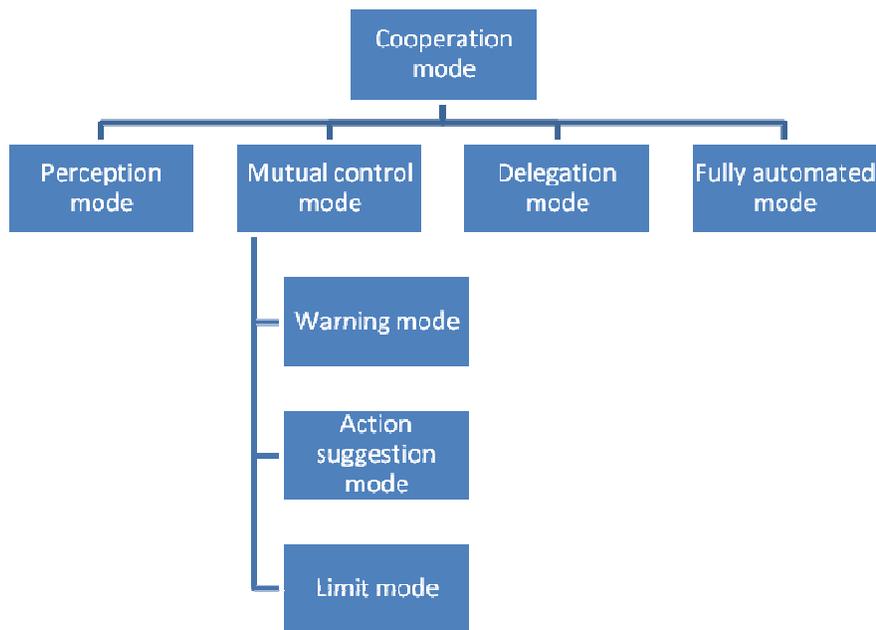
- *Warning mode*: the automation part just indicates that something is wrong with the human agent activity, by means for example of auditory signals
- *Action suggestion mode*: the machine presents together with the warning the indication of the suggested action to be performed, by motor priming for example;
- *Limit mode*: some activities are forbidden, for example by creating pedal or wheel resistance.

The one-but-last mode is the **delegation mode**, corresponding to a lasting function delegation from the human to the machine. Among the well-known drawbacks of such approach, well investigated in literature, there are:

- The loss of expertise due to the lack of exercise of the function delegated to the machine
- The complacency, i.e., the neglecting to gather the information needed for the function fulfilment or to supervise the automated function
- Insufficient situation awareness due to the high level of delegation

These drawbacks are the causes of the difficulty of returning to manual control. The return to manual control is envisioned in emergency situation because the human is considered the best adaptive agent in the team. But this can also be not true, if the human agent is not in a suitable state.

Finally, the fully automated mode can be considered as an extreme case of the delegation mode.



**Figure 13. Modes of Cooperation in Hoc's framework**

The application of framework is currently investigated with reference to the adaptive use case of the Lane Change in the automotive domain, in order to derive the eventual limitation of the framework to be addressed when dealing with adaptation.

## 2.4 Human Operator Models

There are different challenges to face in order to get the human factors into a theory of cognition and, on the contrary, to make a theory of cognition useful to



test human factors. The first challenge has to do with the concept of representation and with the idea that high cognitive processes can be considered without modality, or if a perception-decision-action framework can be preferred, given the constant interaction humans have with the environment. In addition, in order to get the human factors into a theory of cognition the theoretical framework on how to consider the mind and brain interface should be specified. Given these premises the choice of a model able to account for the effects observed in human factor analysis should take into consideration the ability for the model itself to generate predictions that can be tested.

Within HoliDes, different types of Human Operator Models are being developed and/or used. The following lists all Human Operator Models of HoliDes:

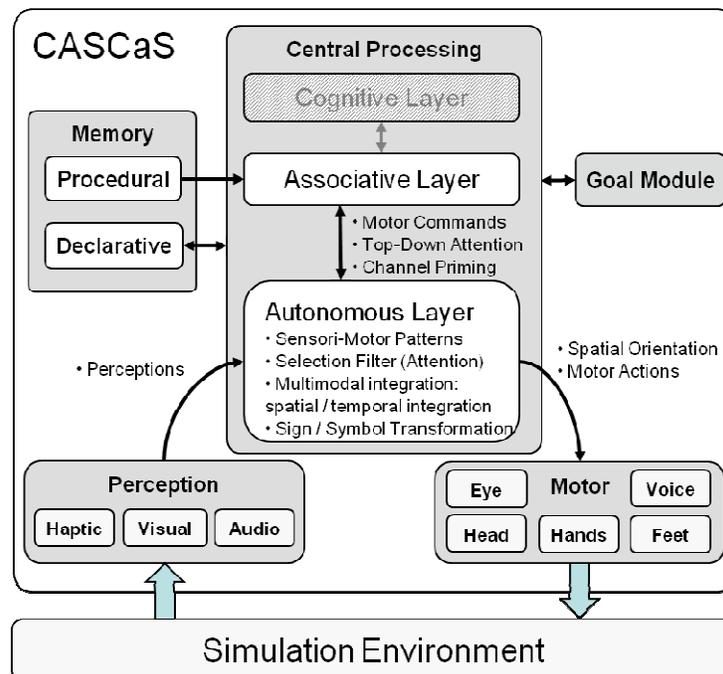
Name	Tool Name	Partner	Domain
CASCaS	CASCaS	OFF	AUT
COSMODRIVE	COSMODRIVE	IFS	AUT
CPM-GOMS	CPM-GOMS	DLR	AUT
DBN driver Model	BadMob	OFF	AUT
Cognitive Driver Distraction Model	Cognitive Distraction Model	TWT	AUT
Pilot Fatigue Model	Pilot Pattern Classifier	TEC	AERO
MDP/MDPN Co-Pilot Model	GreatSPN / Driver Distraction Classifier	UTO, CRF	AUT

In order to capture these different types of models, we started to classify our Human Operator Models. As a first step, we agreed on a set of attributes for classification of our models. As a second step, each model has been classified according to these attributes. The complete classification can be found in Annex IV.

The following sections contain descriptions of the Human Operator models behind the tools described in chapter 3.

### 2.4.1 CASCaS

The Cognitive Architecture for Safety Critical Task Simulation (CASCaS) is a framework for modelling and simulation of human behaviour. Its purpose is to model and simulate human machine interaction in safety-critical domains like aerospace or automotive, but in general it is not limited to those specific domains.



**Figure 14: Structure of the cognitive architecture CASCaS with all internal components and the major data flows.**

Figure 14 shows the current version of the model with all its components. Basically, the architecture consists of 5 components: a *Goal Module* which stores the intentions of the model (what it wants to do next). The *Central Processing* is subdivided into three different layers: the *cognitive layer* which can be used to model problem solving, the *associative layer* executes learned action plans and the *autonomous layer* simulates highly learned behaviour.

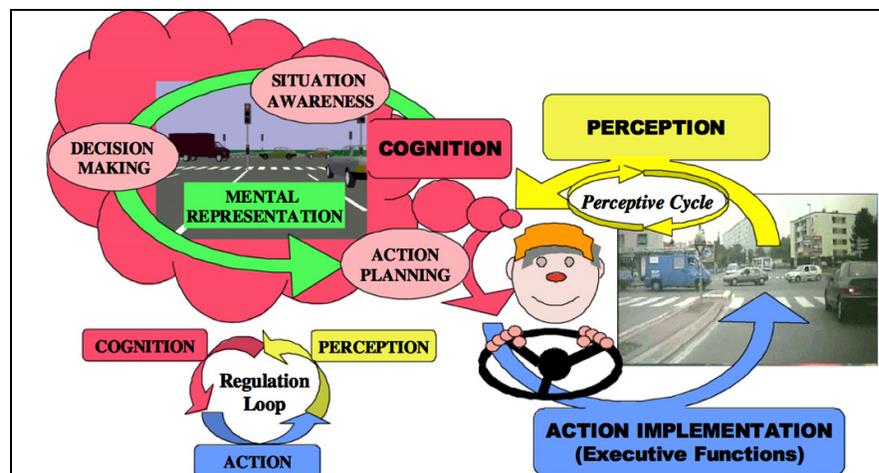
These levels are equivalent to the three behavioural layers of Rasmussen [143] (knowledge based, rule based, skill based). The memory component is subdivided into a procedural (action plans) and a declarative knowledge (facts) part. The *Perception* component contains models about physiological characteristics of the visual, acoustic and haptic sensory organs, for example models about peripheral and foveal vision.

To interact with the external environment the *Motor* component of CASCaS contains models for arm, hand and finger movements. It also comprises a calculation for combined eye / head movements that are needed to move the visual perception to a specific location. In general the model starts observing its environment via the perception and receives input which is stored in the memory component. Depending on its current intention and on the perceived information from the environment, it selects action plans and tries to achieve its current goal. It may generate new goals and further actions, which can be triggered by events

perceived from the environment or the model may itself create new goals, based on its own decision making process, to initiate a certain behaviour.

## 2.4.2 COSMODRIVE

COSMODRIVE is a Cognitive Simulation Model of the car Driver developed at IFS, in order to provide computational simulation of car drivers. The general objective is to virtually simulate the human drivers' perceptive and cognitive activities implemented when driving a car, through an iterative "Perception-Cognition-Action" regulation loop (Figure 15).



**Figure 15: Overview of COSMODRIVE Model theoretical approach**

Through this main regulation loop, the model allows to:

- Simulate human drivers perceptive functions, in order to visually explore the road environment (i.e. *perceptive cycle* based on specific driving knowledge called "schemas"; [16]) and then to process and integrate the collected visual pieces of information in the Cognition Module.
- Simulate two core cognitive functions that are (i) the elaboration of *mental representations* of the driving situation (corresponding to the driver's Situational Awareness; [17]) and (ii) a *decision-making* processes (based on these mental models of the driving situation, and on an *anticipation process* supported by dynamic mental simulations)

Implement the driving behaviours decided and planned at the cognitive level, through a set of effective actions on vehicle commands (like pedals or steering wheels), in order to dynamically progress along a driving path into the road environment.

### 2.4.3 CPM-GOMS

In order to understand and model human behaviour at a small scale, a structured way to organise behavioural data is required. The GOMS framework helps to model a user's behaviour in terms of Goals, Operators, Methods and Selection rules [29]. There are several distinct variants of this approach, with each have different foci [83]. For the application as a modelling framework in the driving domain, we chose CPM-GOMS (Critical Path Modelling GOMS, [81]).

Compared to other tasks typically studied in human-machine interaction driving has unique properties which require an approach that reflects these properties. Among them, time criticality is of most importance. Decisions have to be made in split seconds, and actions often have to be carried out immediately. Second, the environment is rather unstructured, compared to settings in which the task is largely pre-structured by the machine. Further, on a tactical and operational level, drivers typically do not plan their actions a long time ahead. They rather behave opportunistically and seem to be guided more by constraints than a rigid set of rules. Finally, the driving task demands many subtasks and actions to be carried out in parallel.

The approach of CPM-GOMS provides a superb fit to these driving task properties. It frames human behaviour as operators which are carried out by resources. These resources can be motor, perceptual and cognitive, but also system resources could be included. Based on driving data as well as video and eye tracking data we can thus make a description of driver behaviour on a granular time level. Following this, we will derive a cognitive model for the lane change use case from WP9. This model will have the general capability to predict behaviour in specific situations, and may even predict workload at specific points during driving.

Our modelling effort helps designing fluent task transitions between the human driver and a partially automated car, which is IAS' demonstrator. For the design of such a handover-of-control it is critical to understand how this change the driving task and generally where drivers could be best supported with automation.

When designing driver assistance, drivers' needs usually can only be considered on a very general level, such as "minimize workload" or "increase safety". With only very few useful models available that address driving on a dedicated cognitive level (e.g. [147] and [94]), designers and system architects usually have to trust their gut feeling concerning what actually helps drivers accomplish their goals better. During evaluation systems can be only be summarily assessed, and the prohibitive costs of large scale user studies often prevent a deeper understanding of the driver-system-interaction.

Even an informal model of a lane change, such as the one the CPM-GOMS approach provides, informs engineers and Human Factors specialists about what

to look for when designing for an optimal interaction between machine and driver. Potentially, the model can be even formalized and implemented, allowing for the simulation of the human-machine-interaction. This could be done using the model itself, or as input for a model in a cognitive architecture such as Adaptive Control of Thought-Rational (ACT-R). During evaluations of designs, which do not necessarily occur at the end of the design cycle, but are often intermediate steps towards the final design, modelling the human resources, operators and goals helps designing the right experiments in the sense that they inform the Human Factors expert what to look for.

#### **2.4.4 Pilot fatigue model**

The performance of human operator is influenced by his engagement in the activity, which heavily depends on operator's cognitive state. It has been demonstrated that fatigue can cause break-down of the performance and therefore early detection can prevent from critical situations such as attention tunnelling, information missing or misinterpretation.

However the fatigue is a complex psychological phenomenon and its effect on performance has strong subjective component. There is also a strong temporal component as the fatigue develops in time due to varying level of activity and subjective ability of adaptation influences the speed of fatigue onset.

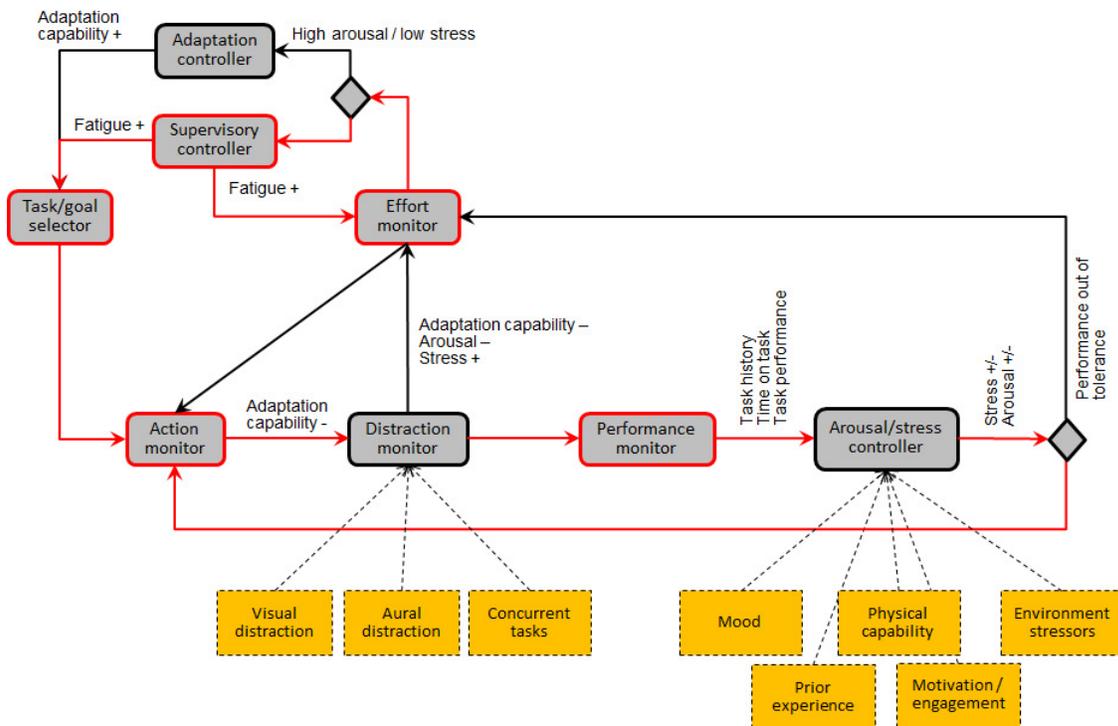
There are a number of models explaining the origin, character and consequences of the fatigue. The hypothesis of compensatory control will be used as the starting point for fatigue modelling in WP7.

The compensatory control hypothesis can be explained by a two level model where the lower level corresponds to the execution of familiar tasks guided by directed attention whereas the upper level refers to the intervention of executive control. The executive control is conceived as a limited resource for controlled processing especially for activities such as planning, problem solving, and task scheduling. The prolonged (over)use of this mechanism is the cause for the phenomena of mental fatigue (Hockey & Earle, 2006).

To model the situation, UML state diagram is used. The variables in the model are the fatigue and the most prominent factors that affect the fatigue – the ability of adaptation as negative feedback decreasing effects of fatigue, the stress as positive feedback, and the arousal as indifferent factor. The states represent cognitive activities that are performed in task execution and which may affect the model variables. Transitions between states are described by transition functions that take into account effects of time, external environment etc. The model is shown in Figure 16, the red components form the original compensatory control model.

The transitions functions describe how situation influences the variables of the model. They take into account time on task and history of previous tasks to address long term development of fatigue. The form of the transition functions will be specified by further modelling and experimental work.

The function of the model is to assess level of fatigue in real time based on directly measurable information. Inputs from both the task analysis and physiological markers will drive the model in time and will determine how often the supervisory controller is used and thus how the fatigue builds up.



**Figure 16: State diagram of compensatory control model extended for temporal properties. In yellow, measurable properties that influence the behaviour of the model.**

### 2.4.5 Cognitive driver distraction model

*Problem to be solved*

Distraction during driving leads to a delay in recognition of information that is necessary to safely perform the driving task. Four different forms of distraction are distinguished while they are not mutually exclusive: visual, auditory, bio-



mechanical (physical), and cognitive distraction. Human attention is selective and not all sensory information is processed (consciously). When people perform two complex tasks simultaneously, such as driving and having a demanding conversation, the brain shifts its focus. This kind of attention shifting might also occur unconsciously. Driving performance can thus be impaired when filtered information is not encoded into working memory and so critical warnings and safety hazards can be missed.

Such a change in attention can be observed by change in behaviour and physiological properties which can be measured (e.g., different reaction times to events, facial movements such as eye-blink). In addition, certain environmental conditions can be assumed to cause distraction (loud music, a crying baby on the back seat, an intense discussion between driver and passengers).

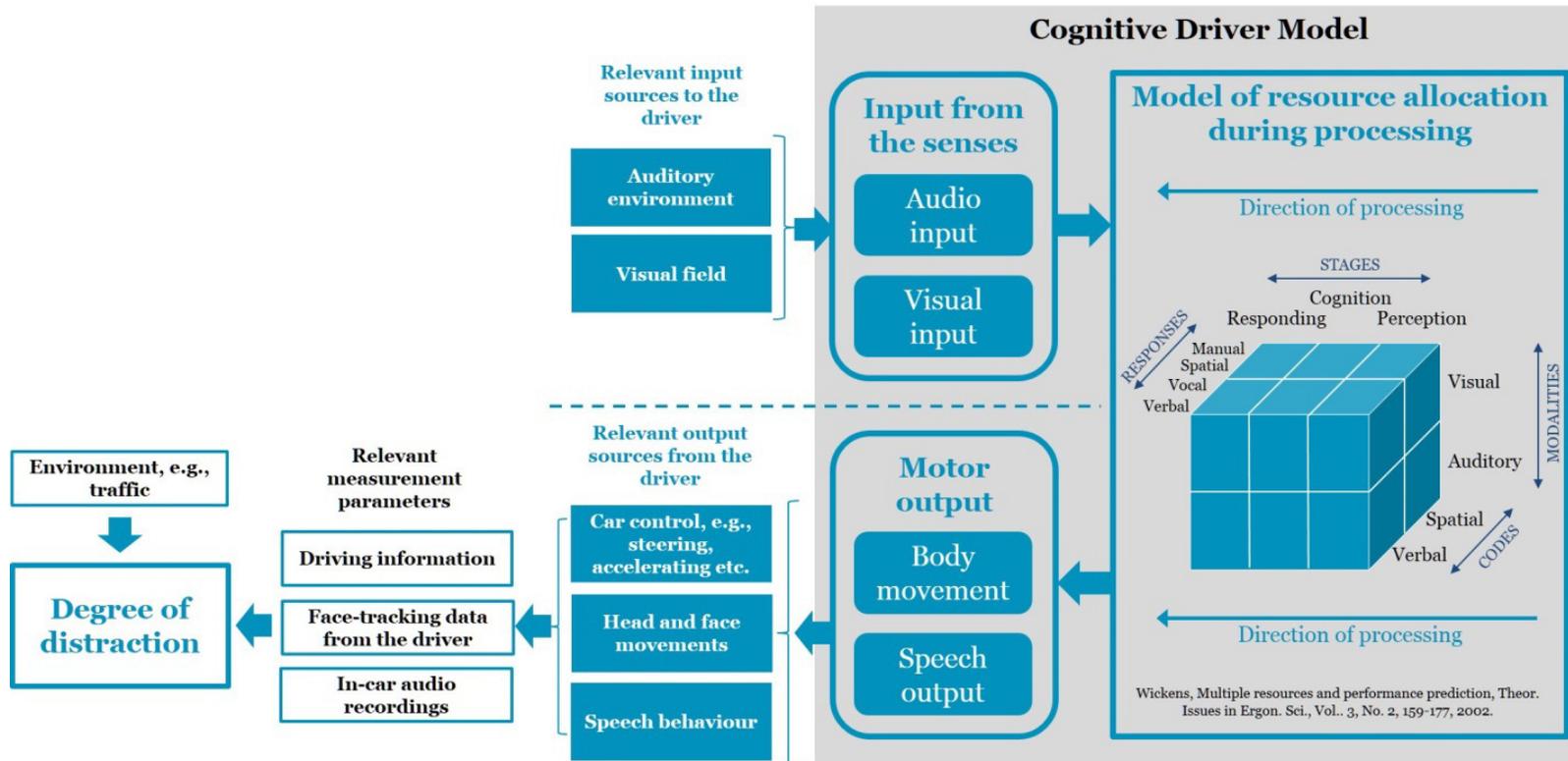
To combine such different clues, a computational and empirical cognitive distraction model will be developed in order to analyse different signals from in-car measures with the purpose to detect the distraction degree of the driver. These measures will include an acoustic analysis including, e.g. the detection of the number of speakers, the degree of emotional content, and information about the driver's involvement in the conversation (e.g., whether the driver himself is speaking). In addition, face-tracking signals such as the blinking of the eyes, head pose and mouth movements will add to the reliability of distraction prediction.



### Model

Figure 17 shows the architectural design of the simplified cognitive distraction estimation model. We take the driver's auditory and visual perception into consideration and compute his/her distraction degree based on a resource allocation model. This model from Wickens (2002) [172] states that the more a secondary task takes up the same or similar sensory modalities (auditory vs. visual), codes (verbal vs. spatial) and processing stages (perceptual, cognitive, responses), the more the secondary task leads to distraction from the primary task. The measured parameters derived from in-car audio recordings, face-tracking information of the driver, behavioural car information (e.g. driving parameters) and environmental information like the distance to the pace car to be followed will lead to conclusions about the allocation of the driver's resources and therefore enable the computation of his distraction degree.

This cognitive model helps us to understand which factors influence cognitive distraction and therefore it helps us focus on the relevant types of data we should measure and select significant features. This knowledge is applied in the computational distraction classification model, which is aimed to analyze the selected data real-time, and provide the interpreted level of distraction as an output. To this end we investigate standard machine learning algorithms (work-in-progress). The cognitive model can be seen as the theoretical framework (developed in WP2), and the classification model as the applied framework (developed in WP5).



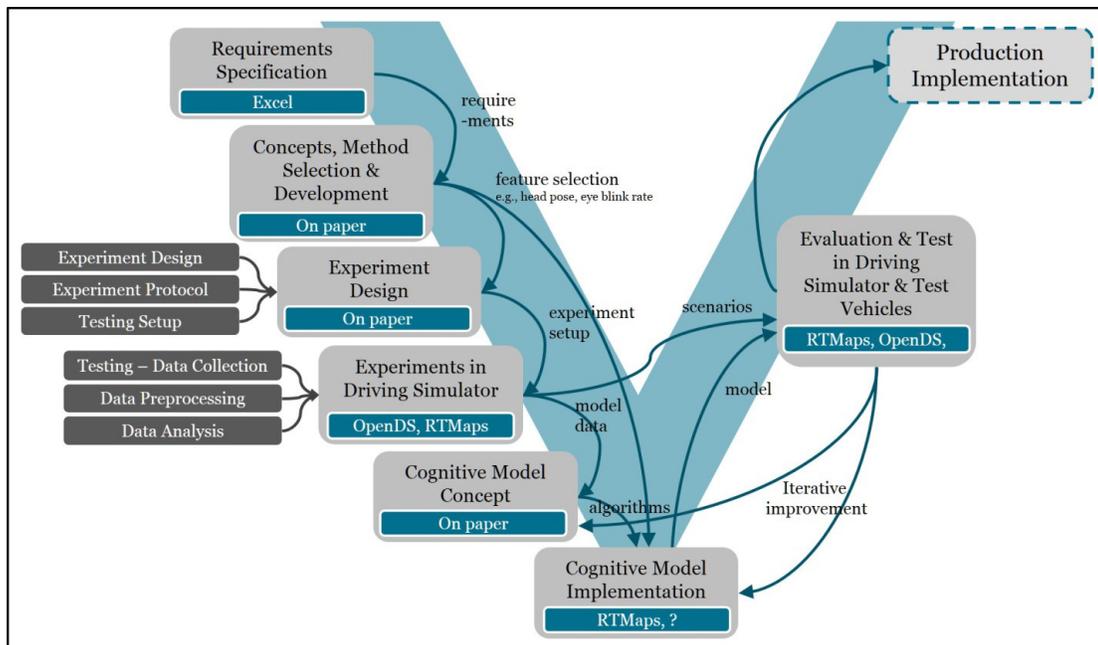
**Figure 17: Architectural design of the cognitive model predicting the driver's distraction degree**

Usage (Development phases)

The distraction estimation model bears the potential to be used online as a tool to classify the driver's distraction not only during testing of a prototype, but also during everyday interaction with the AdCoS. This online measure of distraction could in turn be used to adapt the degree of automation of the AdCoS to the driver's state.

In addition, the model plays an essential role during system validation phase. When validating a system it can be very valuable to derive knowledge about the human while interacting with a prototype or some modules of an AdCoS. The cognitive model can be used here to determine the existing input the human has to cope with. After that, the computational model can be used to provide feedback whether or not a new system (module) increases or decreases the operator's degree of distraction.

Figure 18 shows the individual steps of the workflow integrating the distraction estimation model as well as the MTT into the development of an AdCoS using the HF-RTP. Activities specific for the HF-domain are those related to the experimental design, the testing procedure, data analysis as well as the identification of data predicting the degree of the driver's distraction.



**Figure 18: Architectural design of the cognitive model predicting the driver's distraction degree**

## 2.4.6 The MDP/MDPN Co-pilot model

The driver (co-pilot) model for the CRF demonstrator (also called co-pilot) has as a central core which computes a “driving strategy” that is then suggested to the user through an appropriate, adaptive HMI. The modelling formalism used to describe the driver model is that of Markov Decision Process (MDP) [141], a well-known formalism defined by Bellman in the early sixties for studying optimization problems.

### 2.4.6.1 Markov Decision Processes and Markov Decision Petri Nets

An MDP is a stochastic control process in which, at each time step, the modelled entity is in some state  $s \in S$ , and a decision maker may choose any action  $a \in A$  that is available while in  $s$ . Then, the process goes into a new state  $s'$  according to a specified transition probability (random choice), providing feedback to the decision maker in the form of a corresponding reward (or cost)  $R(a,s,s')$  (depending by the chosen action and by the source and destination state). A key notion for MDPs is the strategy, which defines the choice of action to be taken after any possible time step of the MDP.

Analysis methods for MDPs can compute the strategies that maximize (or minimize) a target function based on the MDP’s rewards (or costs). In this way the MDP model is used to compute the optimal strategy, which is suggested to the human to achieve her/his goal.

The MDP used in the CRF demonstrator presents incomplete or uncertain transition rates; consequently the decision process is optimized with respect to the most robust policy, which corresponds to the best worst case behaviour.

Since MDP is a low level formalism, then it might be difficult to represent directly at this level a complex real system as our AdCoS.

To cope with this aspect we are using Markov Decision Petri Net (MDPN) [14] a higher-level formalisms whose semantic is MDP.

The main feature of MDPNs is the possibility to specify the general behaviour as a composition of the behaviour of several components, some of which are subject to local non deterministic choice, and are thus called controllable, while the others are called non controllable. Moreover any non-deterministic or probabilistic transition of an MDP can be the result of a set of non-deterministic or probabilistic steps, each one involving a subset of components. Hence, an MDPN model is composed of two parts, both specified using the Petri Net (PN) formalism of the classical Place/transition type, extended with priorities associated with transitions: the  $PN^{nd}$  subnet and the  $PN^{pr}$  subnet, describing respectively the non-deterministic (nd) and probabilistic (pr) behavior.

The two subnets share the set of places, while having disjoint transition sets. In both subnets the transitions are partitioned into **run** and **stop** subsets, and each transition has an associated set of components involved in its firing (in the  $PN^{nd}$  only controllable components can be involved). Transitions in  $PN^{pr}$  have a weight

attribute, used to compute the probability of each firing sequence. A non-deterministic or probabilistic transition at the MDP level is the result of the firing of zero or more **run** transitions followed by the firing of a **stop** transition. Moreover, in MDPN a reward/cost function can be specified in terms of state reward/cost, called  $rs()$ , and non-deterministic transition reward/cost, called  $rt()$ . A global reward function is the sum of a state reward function and of an action reward function.

Since it has been decided that the MDP in the CRF demonstrator should contain incomplete or uncertain transition rates, the MDPN formalism has been extended to include uncertainty. In particular uncertainty has been introduced at the level of the transition rates of  $PN^{pr}$ . In this way, the MDPN underlying process becomes an MDP with incomplete or uncertain transition rates. This leads to the following definition.

A Markov Decision Petri Net (MDPN) with uncertain is a tuple  $\langle Comp^{pr}, Comp^{nd}, N^{pr}, N^{nd} \rangle$  where:

- $Comp^{pr}$  is a finite non empty set of components;
- $Comp^{nd} \subseteq Comp^{pr} \cup \{ids\}$  is the non-empty set of controllable components;
- $N^{pr}$  is defined by a Petri net with priorities  $\langle P, T^{pr}, I^{pr}, O^{pr}, H^{pr}, prio^{pr}, m_0 \rangle$ , plus (1) a mapping function  $UWeight: T^{pr} \rightarrow R^2$  that specifies an interval in which the transition rate can vary (uncertainty on the transition rates), and (2) a function  $act: T^{pr} \rightarrow 2^{Comp^{pr}}$  that defines the  $Comp^{pr}$  components involved in the probabilistic transition firing. Moreover,  $T^{pr} = Trun^{pr} \cup Tstop^{pr}$ .
- $N^{nd}$  is defined by a Petri net with priorities  $\langle P, T^{nd}, I^{nd}, O^{nd}, H^{nd}, prio^{nd}, m_0 \rangle$  and a mapping function  $obj: T^{nd} \rightarrow Comp^{nd}$ , that defines the components involved in the non-deterministic transition firing. Moreover,  $T^{nd} = Trun^{nd} \cup Tstop^{nd}$ .

Furthermore, the following constraints must be fulfilled:

- $T^{pr} \cap T^{nd} = \emptyset$ . A transition cannot be non-deterministic and probabilistic at the same time.
- $\forall id \in Comp^{pr}, \exists C \in Comp^{pr}$ , so that  $id \in C$  and  $act^{-1}(\{C\}) \cap Tstop^{pr} \neq \emptyset$ . Every component must trigger at least one final probabilistic transition.
- $\forall id \in Comp^{nd}, obj^{-1}(\{id\}) \cap Tstop^{nd} \neq \emptyset$ . Every controllable component must be the object of at least one final non deterministic transition.

#### 2.4.6.2 MDPN as co-pilot model

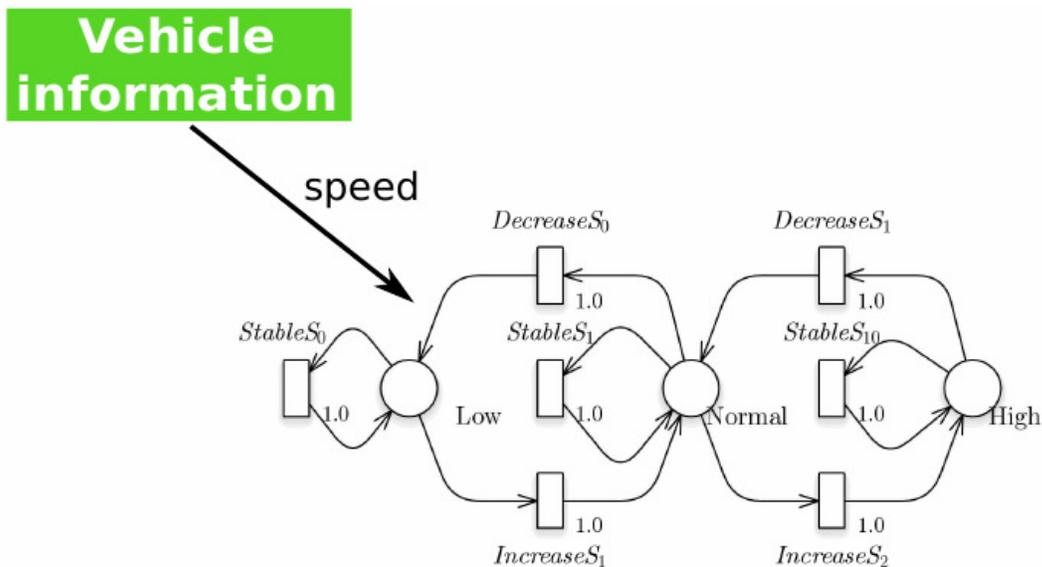
The MDPN model of the co-pilot is still work in progress, but the following system's components have already been identified:

- A **vehicle component** describing the vehicle dynamic status (according to the information available on CAN bus);

- A **driver component** describing the driver status;
- An **obstacle components** describing the obstacles' status in terms of its relative speed and position (e.g. longitudinal and lateral) w.r.t. our vehicle;
- An **action component** describing the possible macro-actions (e.g. to break, to do no action, to send a warning...) that the artificial driver can execute.

It naturally follows that the first three types of components (i.e. vehicle component, driver component, and obstacle components) will be used to generate the corresponding  $N^{pr}$  net (i.e. the net describing the probabilistic behaviour), while the last one the  $N^{nd}$  net (i.e. the net describing the decision phase).

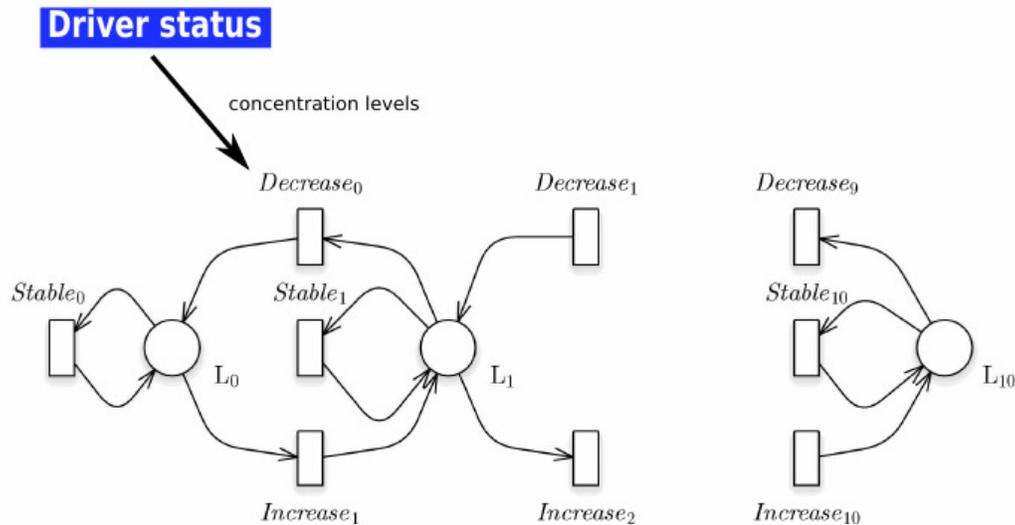
Hereafter we present a preliminary MDPN model for each introduced component.



**Figure 19: MDPN model for vehicle component**

Figure 19 shows an example of MDPN model for the **vehicle component** in which the vehicle speed is explicitly modelled as a discrete variable assuming values: *low, normal, high*.

Probabilistic stop transitions  $StableS_i$ ,  $IncreaseS_i$  and  $DecreaseS_i$  model the speed evolution.



**Figure 20: MDPN model for driver component**

Figure 20 shows the MDPN model describing the **driver's component** according to the possible states identify by the CASCaS module.

In this example we consider ten different levels of driver's attention ( $L_0, L_1, \dots, L_{10}$ ) where  $L_0$  corresponds to the lowest attention level and  $L_{10}$  to the highest one.

Probabilistic stop transitions  $StableS_i$ ,  $IncreaseS_i$  and  $DecreaseS_i$  model how the driver's attention probabilistically during the time.

MDPN model for an **obstacle component** is replicated for each considered obstacle in our case studies. This MDPN model describes the obstacle in terms of its relative speed and distance w.r.t. the vehicle.

In details, as shown in Figure 21, we consider speed and distance as a discrete variables which can assume values: *low, normal, high* for speed, and *collision, close and far* for distance. Speed and distance evolution are modelled by the probabilistic stop transitions:  $StableS_i$ ,  $IncreaseS_i$ ,  $DecreaseS_i$ ,  $StableD_i$ ,  $IncreaseD_i$ ,  $DecreaseD_i$ .

rate of blue transitions depends  
by the speed of the obstacle and our vehicle

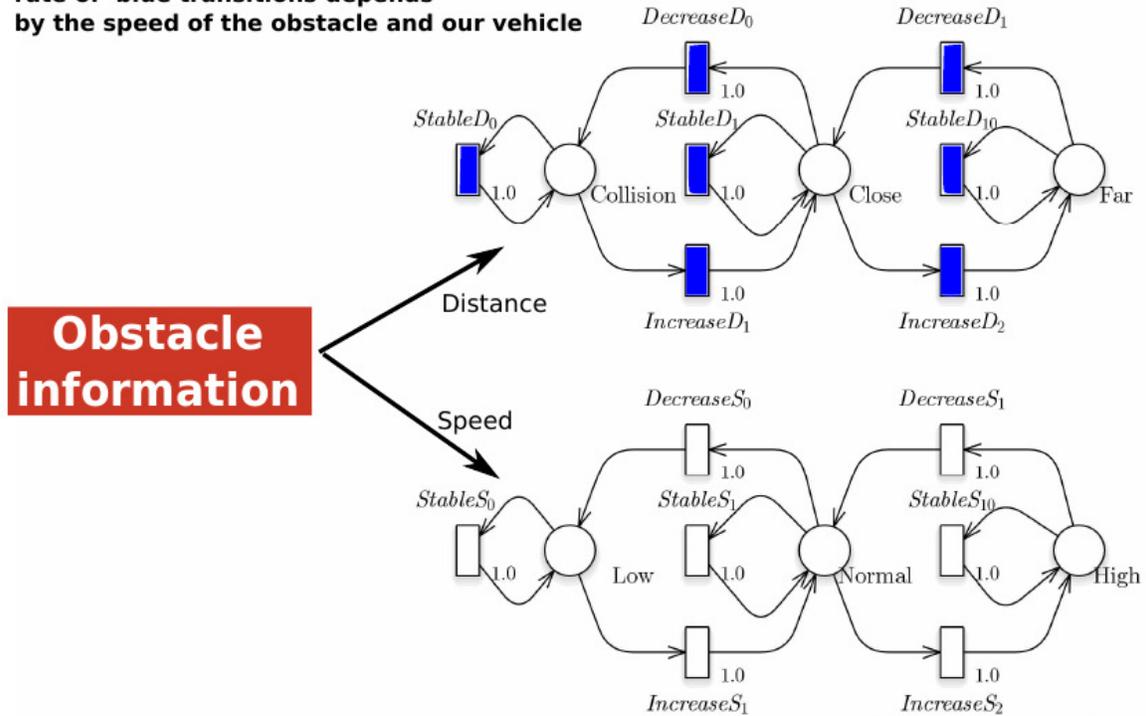


Figure 21: MDPN model for a single obstacle component

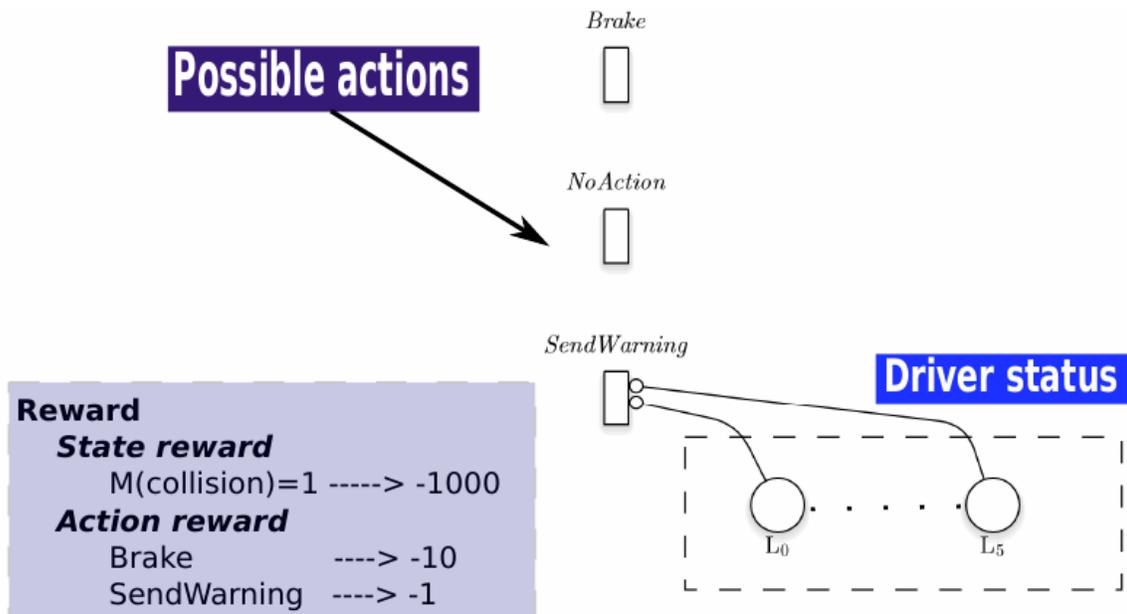


Figure 22: MDPN model for action component



Figure 22 depicts an example of **action component**, the nondeterministic Transitions **Break**, **NoAction** and **SendWarning** represent the possible macro-actions that can be chosen during the decision phase. Observe that the macro-action **SendWarning** is possible only if the driver attention level is greater than  $L_5$ .

The reward function for the MDPN model can be defined by combining the following transition reward:

```

if action Break is selected then it returns CostBreak;
else
    if action SendWarning is selected then
        it returns CostSendWarning
    else it returns 0;

```

with the following marking reward:

```

if place Collision is marked then
    it returns CostCollision
else it returns 0;

```

with  $\text{CostCollision} \gg \text{CostBreak} \geq \text{CostSendWarning}$ .

This obtained reward function is hence able to assure that the system goal is to avoid collision minimizing the total number of actions **Break** and **SendWarning**.

Obviously, more complex reward functions could be also investigated during the project.

## 2.4.7 The DBN driver model

### 2.4.7.1 (Dynamic) Bayesian Networks

When discussing Dynamic Bayesian Networks (DBNs), we will be concerned with probability distributions over sets of discrete and continuous random variables. Variables will be denoted by capital letters, such as  $X, Y, Z$ , while specific values taken by those variables will be denoted by corresponding lowercase letters  $x, y, z$ . The set of values that a random variable  $X$  can take will be denoted by  $\text{Val}(X)$ . We use boldface type capital letters  $X, Y, Z$  to denote sets of random variables (e.g.,  $X = \{X_1, \dots, X_n\}$ ) and corresponding boldface lowercase letters  $x, y, z$  to denote assignments of values to the variables in these sets (e.g.,  $x = \{x_1, \dots, x_n\}$ ). For time series, we assume that the timeline is discretized into time slices with a fixed time granularity. We will index these time slices by non-negative integers and will use  $X_t^t$  to represent the instantiation of a variable  $X_t$  at time  $t$ . A sequence  $X_t^1, X_t^{j+1}, \dots, X_t^k$  will be denoted by  $X_t^{j/k}$  and we will use the notation  $x_t^{j/k}$  for an assignment of values to these sequences. Probability distributions and

conditional probability distributions (CPDs) will be denoted by  $P(\cdot)$ , while probability density functions (PDFs) will be denoted by  $p(\cdot)$ . As the probability  $P(X = x)$  of a single value  $x$  for a continuous variable  $X$  with a PDF  $p(X)$  is always zero, we imply without further notion that each assignment  $X = x$  of a continuous variable  $X$  is replaced by an expression  $x - \epsilon \leq X \leq x + \epsilon$ . When  $\epsilon$  is sufficiently small, the probability  $P(x - \epsilon \leq X \leq x + \epsilon)$  can be approximated by

$$P(x - \epsilon \leq X \leq x + \epsilon) = \int_{x-\epsilon}^{x+\epsilon} p(x) dx \approx 2\epsilon p(x).$$

Using the same  $\epsilon$  for all probabilistic density functions will result in a common pre-factor  $2\epsilon$  in all corresponding expressions that will be canceled during the inference process. That said, we will simply use  $P(\cdot)$  when discussing arbitrary CPDs and PDFs, unless we explicitly want to emphasize that we are dealing with PDFs.

A Bayesian Network (BN) is an annotated directed acyclic graph (DAG) that encodes a joint probability over a set of random variables  $\mathbf{X} = \{X_1, \dots, X_n\}$ . Formally, a BN  $\mathcal{B}$  is defined as a pair  $\mathcal{B} = \{G, \theta\}$ . The component  $G$  is a DAG, whose vertices correspond to the random variables  $X_1, \dots, X_n$ , 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 empty) set of parents  $\text{Pa}(X_i)$  in the graph  $G$ . The component  $\theta$  represents a set of parameters that quantify the probabilities of the BN. Given  $G$  and  $\theta$ , a BN  $\mathcal{B}$  defines a unique joint probability distribution (JPD) over  $\mathbf{X}$ , given by the factorization:

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

DBNs extend BNs to model the stochastic evolution over a set of variables  $\mathbf{X} = \{X_1, \dots, X_n\}$  over time. Note that for DBNs, a variable  $X_i$  without time index does not represent a random variable of the actual JPD, but instead a template variable that will be instantiated at different points in time  $t$ , and each  $X_i^t$  is a variable that takes a value in  $\text{Val}(X_i)$ . A DBN  $\mathcal{D}$  is defined as a pair  $\mathcal{D} = \{\mathcal{B}^1, \mathcal{B}^\rightarrow\}$ , where  $\mathcal{B}^1 = \{G^1, \theta^1\}$  is a BN that defines the probability distribution  $P(\mathbf{X}^1)$  and, under the assumption of first-order Markov and stationary processes,  $\mathcal{B}^\rightarrow = \{G^\rightarrow, \theta^\rightarrow\}$  is a two-slice Bayesian network (2TBN) that defines the 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 with corresponding parameters which defines  $P(X_i^t | \text{Pa}(X_i^t))$ , where a parent  $X_j^t \in \text{Pa}(X_i^t)$  can either be in time-slice  $t$  or  $t - 1$ . The JPD  $P(\mathbf{X}^{1:T})$  over an arbitrary number of  $T$  time-slices is then given by the factorization:

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

A fully specified (dynamic) BN can be used for performing inferences, i.e. answering probability queries about posterior probabilities of variables in the model. A probability query consists of two parts: A subset  $\mathbf{E} \subseteq \mathcal{X}$  of random variables in the model, and an instantiation  $\mathbf{e}$  to these variables, called the *evidence*, and a subset  $\mathbf{Y} \subseteq \mathcal{X}$  of variables in the model called the query variables, with  $\mathbf{Y} \cap \mathbf{E} = \emptyset$ . Inference then denotes the computation of the posterior probability distribution over the values  $\mathbf{y}$  of  $\mathbf{Y}$ , conditioned on the fact that  $\mathbf{E} = \mathbf{e}$ :  $P(\mathbf{Y} | \mathbf{E} = \mathbf{e})$ .

Oftentimes, the set of query variables  $\mathbf{Y} \subseteq \mathcal{X}$  and evidence variables  $\mathbf{E} \subseteq \mathcal{X}$  are already fixed during design time, i.e., the model will be used to only answer a limited set of fixed queries  $P(\mathbf{Y} | \mathbf{E} = \mathbf{e})$ . Especially if the potential factorization  $P(\mathbf{E} | Pa(\mathbf{E}))$  is very complex, we can then opt to not model the JPD  $P(\mathcal{X})$  but instead to provide a model for the conditional JPD  $P(\mathcal{X} \setminus \mathbf{E} | \mathbf{e})$ . When comparing this two possibilities, in general, a model of the joint distribution  $P(\mathcal{X})$  is called a *generative* model, while the model of the conditional JPD  $P(\mathcal{X} \setminus \mathbf{E} | \mathbf{e})$  is called a *discriminative* model.

Modelling DBNs requires the selection and definition of the random variables included in the model, the specification of a graph-structure that specifies the factorization of the JPD of these variables, and the specification of all parameters needed to calculate the probabilities of the (conditional) probability distributions induced by the selected factorization. In general, these steps will be guided in respect to a set of experimental data, which in the case of HoliDes, refers to a set of multivariate time-series of human behaviour traces.

#### 2.4.7.2 DBNs as Human Behaviour Models

Although, described in more detail in Section 3.5.4, the potential use of DBNs as human behaviour models should become apparent, if we loosely define the set of variables used for a human behaviour model and give a brief overview of the kind of probability queries we'd like to answer with them.

For a human behaviour model, we assume the set of variables  $\mathcal{X}$  to consist of a single variable  $I$  representing the *intentions* of a human operator, a single variable  $B$  representing different high-level *behaviours* we expect the human operator to perform, a set of discrete and continuous variables  $\mathbf{A} = \{A_1, \dots, A_n\}$  representing his different *actions* that compose said behaviours, and a set of discrete and continuous variables  $\mathbf{O} = \{O_1, \dots, O_m\}$  representing different *observations* that can be made of the overall cooperative system environment

and the environment the human operator inhabits. For a human behaviour model we then aim to model the evolution of these variables over time, by defining e.g., a generative model with the joint probability distribution  $P(I^{1:T}, B^{1:T}, A^{1:T}, O^{1:T})$  according to a factorization

$$P(I^{1:T}, B^{1:T}, A^{1:T}, O^{1:T}) = \prod_{t=1}^T P(I^t | Pa(I^t)) P(B^t | Pa(B^t)) \prod_{i=1}^n P(A_i^t | Pa(A_i^t)) \prod_{j=1}^m P(O_j^t | Pa(O_j^t)),$$

or a discriminative model  $P(I^{1:T}, B^{1:T}, A^{1:T} | O^{1:T})$  according to a factorization

$$P(I^{1:T}, B^{1:T}, A^{1:T} | O^{1:T}) = \prod_{t=1}^T P(I^t | Pa(I^t)) P(B^t | Pa(B^t)) \prod_{i=1}^n P(A_i^t | Pa(A_i^t)).$$

Given a fully specified human behaviour model, it can be used to constantly (at each time step  $t$ ) infer the joint belief state of intentions and behaviours, given all available evidence about actions and observations observed so far:  $P(I^t, B^t | a^{0:t}, o^{0:t})$ . Given this joint belief state, we can easily obtain the marginal belief states of intentions  $P(I^t | a^{0:t}, o^{0:t})$  and behaviours  $P(B^t | a^{0:t}, o^{0:t})$ . The estimation of the belief state is known as filtering and can be solved by in constant time by recursively computing  $P(I^t, B^t | a^{0:t}, o^{0:t})$  from the past belief state  $P(I^{t-1}, B^{t-1} | a^{0:t-1}, o^{0:t-1})$ .

### 2.4.7.3 Modelling Language

DBNs are not restricted to any specific application or domain and there is no apparent benefit to arbitrarily restrict the scope of the modelling language to human behaviour models. Consequently, our modelling language will cover all DBNs that satisfy the assumption of a first-order Markovian system (see 3.5.4.3.2.1). As the exact formulation of the meta-model is still work in progress, we will focus on the information that an instance of such model must provide, which form a natural three-level hierarchy of abstraction:

1. The definition of all variables in the model  $\mathcal{X} = \{X_1, \dots, X_n\}$ .
2. Under the assumption of a first-order Markovian system (3.5.4.3.2.1), a factorization for the initial time-slice  $P(\mathcal{X}^1 | \mathcal{Y}^1)$ , where  $\mathcal{Y}^1$  may denote the empty set of parents

$$P(\mathcal{X}^1 | \mathcal{Y}^1) = \prod_{i=1}^n P(X_i^1 | Pa(X_i^1)),$$

and a factorization for the 2TBN  $P(\mathcal{X}^t | \mathcal{X}^{t-1}, \mathcal{Y}^t)$ , where  $\mathcal{Y}^t$  may denote the empty set of parents

$$P(\mathcal{X}^t | \mathcal{X}^{t-1}, \mathcal{Y}^t) = \prod_{i=1}^n P(X_i^t | Pa(X_i^t)).$$

3. A set of parameters  $\theta_{x,Pa(x)}$  for each factor  $P(X|Pa(X))$  that is sufficient to identify a function  $f(x, pa(x), \theta_{x,Pa(x)}) = P(x|pa(x))$  that given  $pa(x)$  and  $x$ , returns the conditional probability  $P(x|pa(x))$  according to the parameters  $\theta_{x,Pa(x)}$ .

This compact representation can easily be captured in different meta-model languages e.g., in UML, XML, or Ecore.

The expressive power of these models depends on the set of functions  $f(x, pa(x), \theta_{x,Pa(x)})$  that the meta-model provides, and in order to utilize such a model as a computational model, we need an inference engine that supports these functions. Under the assumption that a given inference framework supports all functions needed for the model, in general, one can easily implement an interpreter that transform the meta-model instance into a computational model usable in the selected framework.

In the following, we will give a brief overview over the kind of functions  $f(x, pa(x), \theta_{x,Pa(x)})$  we need for human behaviour models in HoliDes. As these models consist of both discrete and continuous variables, the following dependencies can occur:

1. A discrete variable with discrete (or the empty set of) parents
2. A discrete variable with continuous parents
3. A discrete variable with discrete and continuous parents
4. A continuous variable with discrete (or the empty set of) parents
5. A continuous variable with continuous parents
6. A continuous variable with discrete and continuous parents

In the first case, we can represent each possible dependency by a tabular representation of the CPD, described in Section 2.4.7.3.1.

Concerning the second and third case, by now, we prohibit the direct dependence of discrete variables by continuous (potentially combined with discrete) parents, we do however allow an alternative representation, described in Section 2.4.7.3.5 that allows to reformulate such dependencies.

For the fourth case, we restrict our modelling language to Gaussians, described in Section 2.4.7.3.2, and mixture of Gaussians, described in Section 2.4.7.3.3. For the last two cases, we restrict our modelling language to conditional linear Gaussians, described in Section 2.4.7.3.4.

#### 2.4.7.3.1 Conditional probability tables (Table-CPDs)

Let  $X$  be a discrete variable, and the set of parents  $Pa(X)$  be composed of only discrete variables (including the empty set of parents), we can represent any CPD  $P(X|Pa(X))$  by a table of probabilities  $P(x|pa(X))$  for each combination of

$x \in X$  and  $pa(X) \in Pa(X)$ . Such a table can easily be specified by a set of parameters  $\theta_{x,pa(x)}$  with an entry  $\theta_{x,pa(x)} = P(x|pa(X))$  for each combination of  $x \in X$  and  $pa(X) \in Pa(X)$ .

#### 2.4.7.3.2 (Multivariate) Gaussians

When dealing with continuous variables, CPDs cannot be represented by a table and we must resort to parametrical families of PDFs. Gaussian distributions are the most commonly used parametric form for continuous density functions [92]. The Gaussian PDF for a continuous variable  $X$  is fully specified by a mean  $\mu$  and a variance  $\sigma^2$ , and given by:

$$p(x) = \mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right].$$

When conditioned by a set of discrete parent variables  $\mathbf{U} = \{U_1, \dots, U_m\}$ , we need to specify a set of parameters  $\theta_{x,pa(x)}$  with different means and variances for each combination of parent values  $pa(X) \in Pa(X)$ :

$$p(x|pa(X)) = \mathcal{N}(x|\mu_{pa(x)}, \sigma_{pa(x)}^2) = \frac{1}{\sigma_{pa(x)}\sqrt{2\pi}} \exp\left[-\frac{(x-\mu_{pa(x)})^2}{2\sigma_{pa(x)}^2}\right].$$

In the case of a more than a single continuous variable, the multivariate Gaussian is the most widely used joint probability density function. A multivariate Gaussian distribution over  $n$  variables  $\mathbf{X} = \{X_1, \dots, X_n\}$  is commonly characterized by an  $n$ -dimensional mean vector  $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n)$  and a symmetric  $n \times n$  covariance matrix  $\boldsymbol{\Sigma}$ . Let  $\boldsymbol{\Sigma}^{-1}$  denote the inverse covariance matrix and  $|\boldsymbol{\Sigma}|$  denote the determinant of  $\boldsymbol{\Sigma}$ , the pdf is defined as:

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right].$$

Conditioned by a set of discrete parents, a different set of mean vectors and covariance matrices must be specified for every combination of parent values.

#### 2.4.7.3.3 (Multivariate) Gaussian mixtures

As not every continuous PDF can be reasonable approximated by a Gaussian, we extend our modelling language by finite Gaussian mixtures, which allow to compose a single density function by a finite number of Gaussian components. Given a large enough number of component densities, each arbitrary PDF can be approximated by a mixture of Gaussians [92]. A mixture of Gaussians over a

single continuous variable  $X$ , composed of a finite number of  $n$  Gaussians, defines a density

$$p(x|\Psi) = \sum_{i=1}^n \lambda_i \mathcal{N}(x|\mu_i, \sigma_i^2),$$

where  $\Psi = (\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2, \lambda_1, \dots, \lambda_{n-1})$  denotes a vector of parameters consisting of  $n$  means,  $n$  variances and  $n - 1$  mixing proportions, with  $\sum_{i=1}^n \lambda_i = 1$  and therefore  $\lambda_n = 1 - \sum_{i=1}^{n-1} \lambda_i$ . Correspondingly, a multivariate Gaussian mixture is defined as

$$p(\mathbf{x}|\Psi) = \sum_{i=1}^n \lambda_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i),$$

where  $\Psi = (\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_n, \lambda_1, \dots, \lambda_{n-1})$  denotes a parameter vector consisting of the  $n$  mean vectors,  $n$  covariance matrices and  $n - 1$  mixing proportions. Once again, when conditioned by a set of discrete parents, a different set of parameters must be specified for every combination of parent values.

#### 2.4.7.3.4 Conditional linear Gaussians

In the case of a continuous variable  $X$  with continuous parents  $\mathbf{Pa}(X)$ , we restrict the resulting PDF to *linear Gaussian CPDs* [92]. Let  $\mathbf{Pa}(X) = \{Y_1, \dots, Y_n\}$ , a linear Gaussian CPD is defined by a set of parameters  $\beta_0, \dots, \beta_n$  and a variance  $\sigma^2$  with the PDF:

$$p(X|y_1, \dots, y_n) = \mathcal{N}(x|\beta_0 + \beta_1 y_1 + \dots + \beta_n y_n, \sigma^2).$$

Conditioning on further discrete parents, then requires a different set of parameters for each combination of parent values.

#### 2.4.7.3.5 Alternative representation of CPDs

Especially when dealing with discriminative models we can encounter e.g., CPDs over continuous variables with a large number of continuous and discrete parents that cannot be reasonable approximated by conditional linear Gaussians, or CPDs over discrete variables with (a large number of) continuous and discrete parents. In these cases, the basic definition of conditional probability [92] provides an alternative approach to represent a CPD  $P(X|\mathbf{Y}, \mathbf{Z})$ :

$$P(X|\mathbf{Y}, \mathbf{Z}) = \frac{1}{\sum_{x \in X} P(x, \mathbf{Y}|\mathbf{Z})} P(X, \mathbf{Y}|\mathbf{Z}).$$



This allows us to approximate the complex dependencies stated in  $P(X|Y,Z)$  by an alternative simpler factorization of  $P(X,Y|Z)$  (e.g.,  $P(X,Y|Z) = P(X|Z)P(Y|X,Z)$ ), i.e., we can represent CPDs with a large number of parents as a distinct conditional Bayesian networks and try to approximate the conditioned JPD by a factorization with additional implied independencies. Furthermore, this representation allows using conditional parents for discrete children. Furthermore, this reformulation allows us to use different factorization of  $P(X,Y|Z)$  for different values of  $Z$ , which allows to encode context-specific independencies [92].

## 2.5 HMI Interaction Models

### 2.5.1 The problem of HMI programming

For many years specialized programming techniques have been used for graphical interactive software running on Personal Computers, wherein input were essentially performed using a mouse and a keyboard. Nowadays, the diffusion of new human input and output techniques, smartphones, tablets, network connections and connected objects has widely increased the number of possible combinations for designing interactive software. For example, on tablet computers, inputs can now be entered using a touch-sensitive surface, a connected object such as an air pointer, and internal sensors such as a gyro or an accelerometer. More complex interactive applications can be composed of multiple interactive software components running in a computer, in the firmware of a touch-sensitive table top display, in an internet server, and in multiple sensors across the world. It thus becomes necessary to provide programming techniques that encompass interactive software more widely.

Software components are collections of instructions that can be developed independently and assembled to produce software products. The interoperability of software components is the ability of two or more software components to be interconnected and function properly together. Components are interoperable when there is a syntactically correct way to combine them without adding an adaptation layer, and when their semantics are directly compatible. Interoperability is a major concern in the development of software, because it dictates how software components can be reused and adapted across multiple applications, and when components can be interchanged during the process of designing an application. Interoperability is also a favourable condition for innovation, because it allows connecting components in ways that had not previously been used. For example, driving the position of graphical objects on the display of a tablet with the orientation of the said tablet becomes possible when the accelerometer is made interoperable with graphics and interchangeable with the touch area. Interoperability and interchangeability can also be exploited during the execution of programs, producing connections that programmers do not need to describe explicitly and exhaustively. For instance, a game can be programmed to change randomly during a session which input device the user



must use to control an object, or which transformation law is applied to the input.

Traditional programming languages, such as C, C++, Lisp or Java, have been derived from programming languages focused on computation, by adding features that favour interoperability. For example, functional programming languages define functions as the foremost category of software component, and even treat data variables as functions with no arguments. This recursive architecture facilitates the creation of interoperable components in software where the role of each individual component is to implement a part of the computation of a global result. Similarly, by gathering computation and data in objects, object-oriented languages facilitate the interoperability of components in software where each individual component must store data in order to contribute to the global computation. Object-oriented languages also favour interoperability and reuse by supporting class inheritance. For more complex situations, Design Patterns have been proposed as additional methods for interconnecting software components whose relationships are incompletely described by function calls or inheritance relations.

Interactive software differs from computation-oriented software in several ways that impact software architecture. In terms of execution, computation programs have a start and an end, and execution consists of steps and loops toward the end. In contrast, interactive software waits for inputs and triggers reactive behaviours or computations depending on the inputs received. Interactive software also differs in terms of data management. Maintaining component state and data values is a central concern in interactive software, whereas it is often considered as a side effect in computation software. Interactive software also exhibits a wider variety of how software components are combined. In computation-oriented software, the relation between a function and its arguments has been proved as a sufficient means of combination for most situations. Alternatively, imperative programming languages provide a few control structures (sequence, loops and tests) that can be used to interconnect programming instructions in computation programs. In interactive software, a large number of additional situations can be present. For instance, graphical components can be grouped in scene graphs, animations can be organized to be executed in parallel, graphical objects can be associated to the various states of dialogue components, instructions can be defined to as to be executed when an external event occurs, the visual properties of a graphical object can be defined to vary continuously with the values of data measured in the physical environment.

Traditional programming languages have received extensions to support the execution of interactive software. For example, waiting functions support execution control by external inputs, and threads support parallel execution of actions. With these extensions, they theoretically support the development of interactive software. However, the increase of possible inputs, states and



combinations of components dramatically increases the number of possible executions of a given application. If this multiplicity of possible executions is programmed using the usual control structures, software complexity increases: any modification of the program behaviour requires changes in multiple components, thus restraining the ability to make choices after the initial design phase.

Along with this increase in software complexity, the interoperability of software components tends to decrease, and software development and validation become long, costly and prone to errors. It also becomes difficult to analyse the properties of software at the appropriate level of abstraction, and only certain classes of interactive software can undergo the software certification processes required in some industrial fields. It also becomes difficult to design programming tools that facilitate software development, because there are no visual representations that appropriately capture the structure of software.

Various software patterns have been proposed to reduce the complexity of interactive software developed with traditional programming languages. Each pattern addresses one cause of complexity. The most common software pattern is the call back function and its variants such as the Inversion of Control pattern and the Signal/Slot pattern, which are aimed at limiting the complexity induced by external control. In this pattern, a programmer can register a given function to that it is called when some conditions are met, such as the occurrence of a given type of external input. In some implementations of this pattern, the call-back function is passed a data structure named "event" that contains the information about what caused the call.

Various software patterns have been proposed to curb software complexity by organizing software components according to their roles and defining how they can be combined. For example, with the Model-View-Controller pattern, application components are made of three sub-components that are respectively in charge of managing the data and the computation, visualizing the data, and managing user input. The Presentation-Abstraction-Control and Model-View-View Model patterns have similar structures. Extended scene graphs are another class of patterns, derived from graphical scene graphs, in which various kinds of non-graphical software components can be added as nodes of the graph, so as to align the software architecture of the application on its graphical structure.

Other patterns have been proposed to organize control flows in interactive software, and compensate the limitations of control structures provided by programming languages. For example, Harel [[58]] proposes Statecharts, hierarchical state machine components that can be combined to describe interactive systems. Myers [128] describes a state machine component that can be adapted to program interaction in various kinds of software components. Transitions between states are performed at the occurrence of certain events, and the appearance and behaviour of software depends on said state of software. Dragicevic et al. [[39]] proposes a data-flow system that can be used



to program input management. Nigay et al. [131] describes a multimodal fusion pattern for combining events and states from multiple inputs. Sottet et al. [155] proposes a pattern for managing the adaptation of software to changes in the computing platform and the execution context.

However, each of these solutions addresses only one cause of complexity, and in most interactive software they need to be combined to address all the causes. This constitutes a source of heterogeneity in the structure of software, because these patterns are not interoperable and components created with them are neither interoperable nor interchangeable. For example, value changes in a data-flow system cannot be directly used as an event in a call-back system or a transition in a state machine. Adaptation code must be written to combine them, using the basic mechanisms provided by each programming language, and this introduces additional heterogeneity. This is unsatisfactory in terms of interoperability and introduces new complexity, with all the consequences described earlier.

Partial solutions have been proposed to make these software patterns interoperable. For example, Chatty et al [30] proposes a method for combining state machines and data flows, in which the configuration of data flows changes when state changes. Appert et al [11] proposes another method for combining state machines and data flows, using Java code to perform the adaptation. Elliott et al [44] proposes Functional Reactive Programming, an alteration of the execution semantics of functional languages that allows exploiting the same syntax for expressing both traditional computation and data flows. Chatty et al [31] discloses an application of extended scene graphs for assembling graphics and heterogeneous behaviour components in a homogeneous fashion.

None of the above solutions guarantees that any software application can be created using a single set of homogeneous and interoperable components. In addition, most of these solutions are dedicated to graphical interactive software, and none are extensible enough to introduce new control structures as required by new interaction modalities and new interaction styles. All require the use in programs of instructions from a traditional programming language that provide missing control structures, architecture patterns, or even functionality, with all the consequences described earlier in terms of complexity, interoperability, reuse, certification, etc.

Dedicated languages have been proposed to program classes of interactive software using homogeneous components. For example, the XUL, XAML and QML languages propose recursive architectures for assembling graphical components in user interfaces. However, they cannot easily be extended to other uses than graphical user interfaces, they provide a limited range of control structures, and the applications and interactions that can be produced with them are stereotyped. Producing non-WIMP (windows, icons, menus, pointing)



applications with them requires the use of a general-purpose language, and they cannot be used as general-purpose solutions for interactive software.

Synchronous data flow languages have been created to support the creation of interactive software such as automatic control systems. N. Halbwachs et al. [56] describes a synchronous dataflow language, LUSTRE. Extensions to LUSTRE have been developed to implement user interfaces. In LUSTRE inputs are used for controlling data flows. In addition, LUSTRE code can be used to define state machines. However, the interoperability between state machines and data flows in LUSTRE is limited as in previously described solutions. In addition, it is very difficult to replace one data flow with another, once it is defined. The definition of new control structures is not supported.

For HoliDes, the description of the HMI Interaction model together with the structure of the model is shown in Figure 23.

### 2.5.2A new event-based approach

Since several years, ENA has been developing a general framework (named djnn, described below) dedicated to the development of interactive systems. In the Task 2.5, we extended this framework to increase its expressive power (support of many input sources, displays, etc.) and we complemented it with an XML support. This means that we added the possibility to write a program in pure XML either directly through a text editor or by serializing an existing compiled program.

djnn (available at <http://djnn.net>) is a general framework aimed at describing and executing interactive systems. It is an event driven component system with:

- A unified set of underlying theoretical concepts focused on interaction.
- New architectural patterns for defining and assembling interactive components.
- Support for combining interaction modalities.
- Support for user centric design processes (concurrent engineering, iterative prototyping).

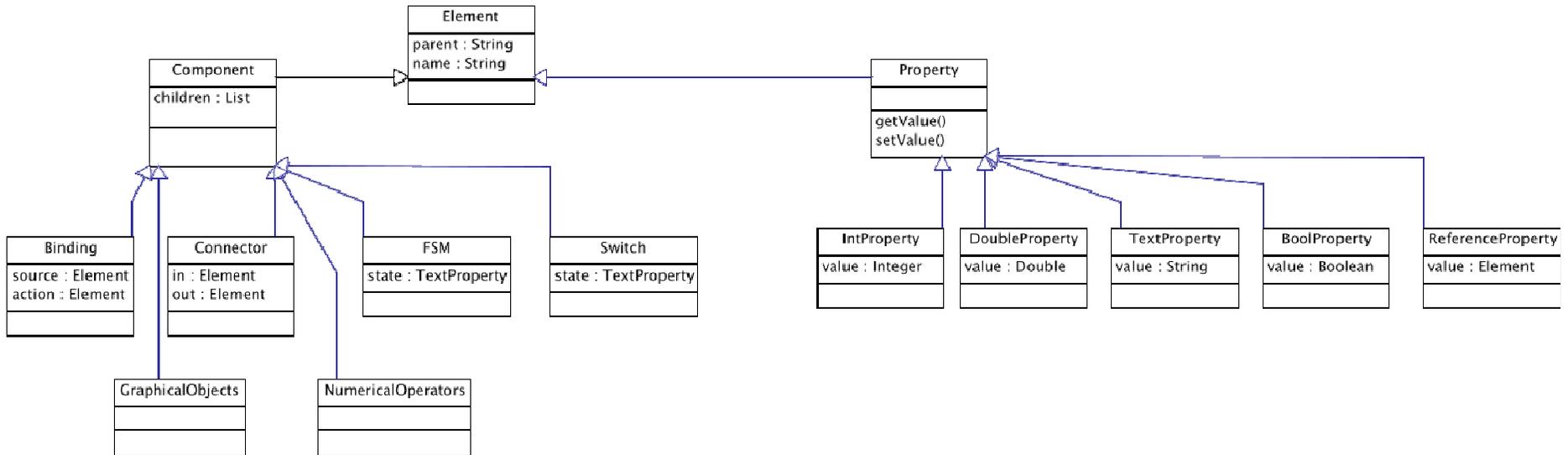
In djnn, every entity you can think of, abstract or physical, is a component (Figure 23). In addition to control structures (binding, dataflow connector, finite state machine), djnn comes with a collection of basic types of components dedicated to user interfaces: graphical elements, input elements (mouse, multi-touch, sensors, etc.), file elements etc. Every component can be dynamically created or deleted.

Moreover, they offer an interface carrying out both generic and specific services:

1. Generic services: all components can be ran or stopped
2. Specific services: element dependent. For example, a graphical element such as a rectangle offers its current position, its width and height, the

	<p style="text-align: center;"><b>HoliDes</b> <b>H</b>olistic Human Factors <b>D</b>esign of Adaptive Cooperative Human- Machine Systems</p>	
--	--	---

size of round corner (horizontal and vertical), mouse pressed event with position, mouse released event, mouse moved event with position, mouse enter event and mouse leave event.

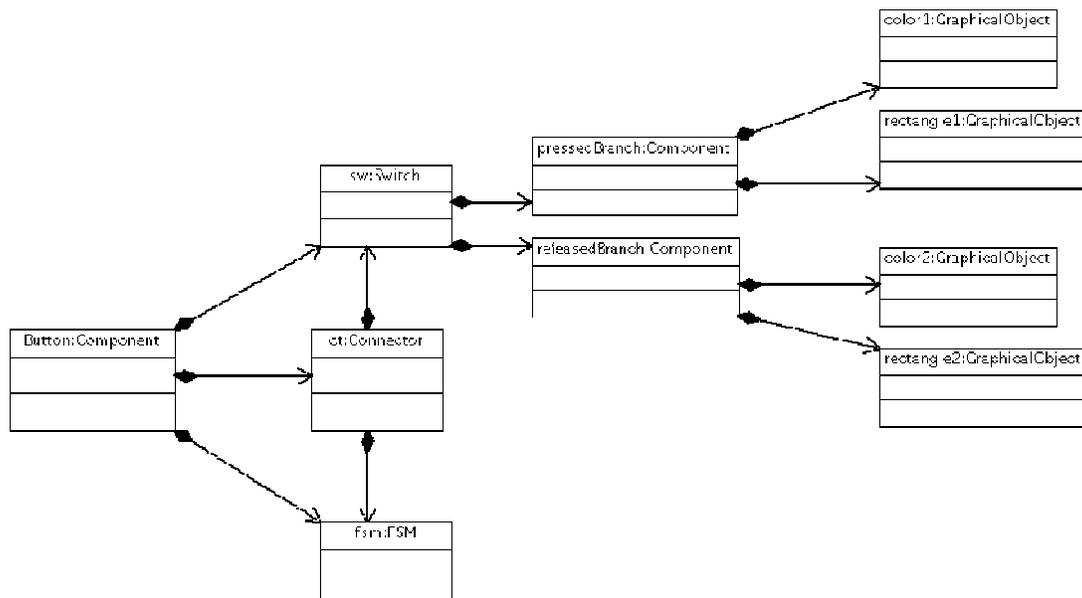


**Figure 23 Simplified UML representation of the component hierarchy**

To design various and large interactive systems, components must be interconnected. For this purpose, two mechanisms are available in the framework:

- Recursive composition: components can contain other components. For example, a complex graphical scene is composed of several graphical sub-components; a mouse is made of two buttons and one wheel; a Finite State Machine (FSM) is made of several bindings etc. The designer can explicitly manage this tree-oriented architecture.
- Transversal connection: all the available components can be connected by control primitives, whatever is their place in the tree of components. For example, a binding can connect a mouse press to a rectangle horizontal position.

Combining FSMs by coupling their transitions, or by controlling the activation of one by a state or a transition of another, makes it possible to create complex behaviours (Figure 24). It also makes it easier to structure applications as collections of reusable components.



**Figure 24: UML representation of a simple djnn button**

In the first year of this project, an abstract syntax and a grammar for djnn have been defined through various XML schemas. The model addresses most components available in djnn, particularly control primitives.

For example Figure 25 contains the description of a binding and a FSM: a binding is an extension of a component containing identification of a source ("trigger") and of a target ("action"). A FSM is an extension of a component containing a sequence of minimum of two states and a sequence of a minimum of one transition (state and transition are defined elsewhere in the XML schema).

```

<xs:complexType name="binding">
  <xs:complexContent>
    <xs:extension base="cmn:core-component">
      <xs:attribute name="source"
        type="xs:string" use="required" />
      <xs:attribute name="action"
        type="xs:string" use="required" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="fsm">
  <xs:complexContent>
    <xs:extension base="cmn:core-component">
      <xs:sequence>
        <xs:element name="state"
          type="state"
          minOccurs="2"
          maxOccurs="unbounded" />
        <xs:element name="transition"
          type="transition"
          minOccurs="1"
          maxOccurs="unbounded" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

**Figure 25: XSD definition of two djnn components**

The main advantages provided by these definitions are:

- Definition of a well-defined model for djnn: illicit constructs using the language can easily and automatically be detected during edition of the model thanks to the XML schema.
- Improvement of interoperability: this evolution is a first step toward the definition of a better integrated tool chain with the capability to dump a concrete graphical user interface (GUI) in an XML file and conversely to load and to execute a GUI from an XML based description.
- To provide a model ready for formal verification. Indeed, given that numerous properties of a system are mirrored in the structure of the tree of its XML representation, it becomes possible to investigate such properties with dedicated tools such as XPath requests.

The next steps are mainly the improvement of the XML support (today, we just have a beta version, not yet publicly available) and to foster the connection with the formal verification tools of the WP4.

## 2.6 Training Models

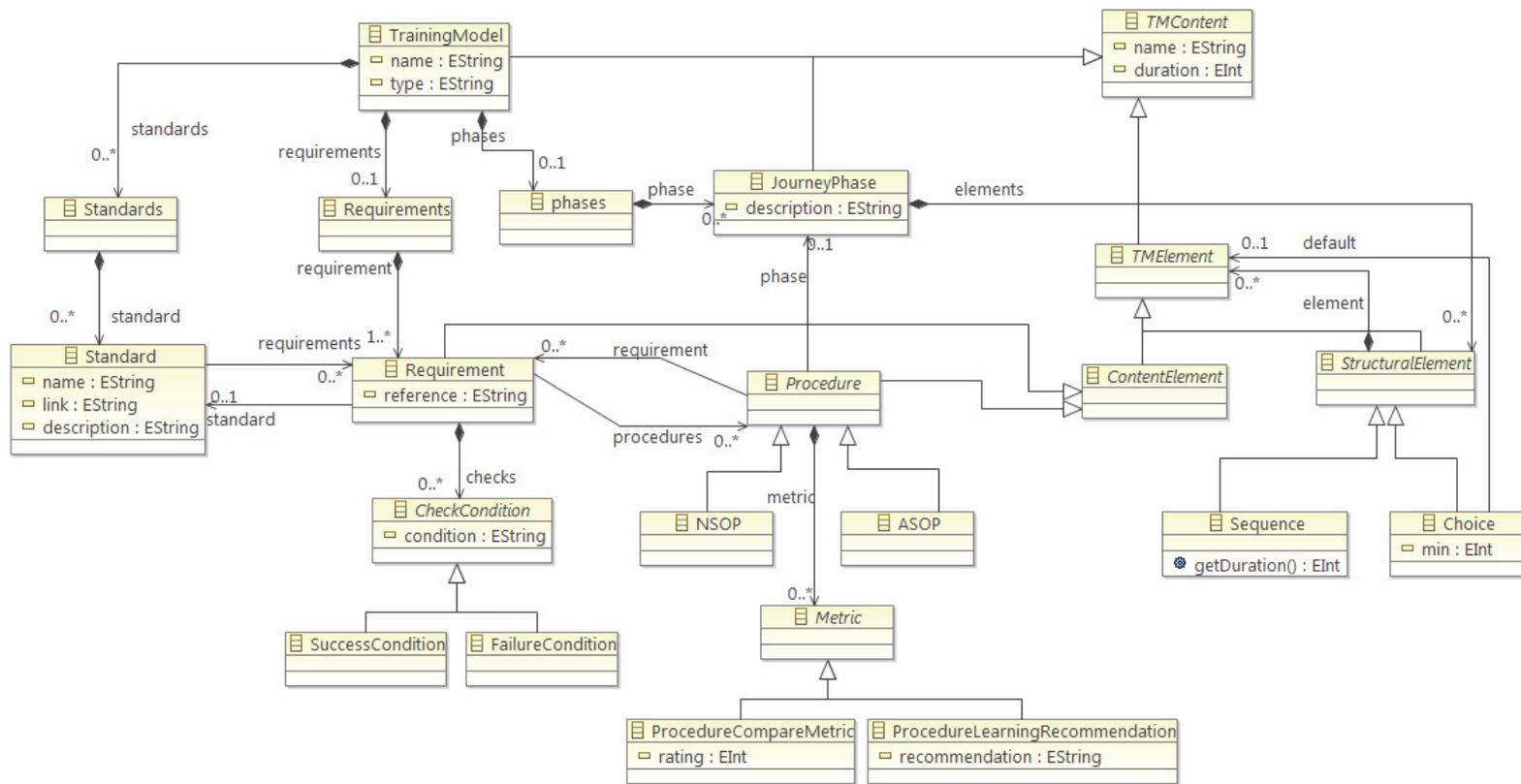
The first version of the training model has been specifically derived for the WP7 Training AdCoS, where the model is evaluated. In future versions, this model will be generalized to other domains.

Figure 26 shows the initial model for training application. Main class is the **TrainingModel** itself, which owns a set of standards the training is related to, a set of requirements the training has to fulfil, as well as a set of phases describing the phases of the journey the vehicle take (e.g. starting the engine, takeoff, climb, cruise, ...).

Each **Standard** describes a source for training **Requirements**, or in other words formal training objectives that the trainee is required to know and apply after the training. For example, a driver is required to know how to operate the gear change (economically), or a pilot is required to know how to start the engines of the aircraft. The requirements are usually part of a check, where the requirements are officially tested before a licence is issued. Therefore, a set of **CheckCondition's** can be assigned to the requirement, which describes conditions for failing or passing the check.

Each requirement is associated with a **Procedure**, which links to a task model (see section 2.1) describing the tasks that are usually needed to reach the requirement successfully. A procedure is broken down into normal procedures (NSOP) performed in standard operation of the vehicle, and abnormal procedures (ASOP), which are performed in abnormal situations, i.e. when a system malfunction occurs. A procedure can have a **Metric**, which holds the result of a certain analysis for that procedure, i.e. a comparison with other procedures (**ProcedureComparisonMetric**).

In addition, the procedure is assigned to a (journey) phase, in which the procedure is usually applied. For each **JourneyPhase**, one can specify which elements must be trained in this phase, i.e. often certain procedures are only applied in a certain phase (e.g. starting the aircraft engines on ground during pre-flight parking), and some malfunctions can only occur during a certain phase (e.g. the ignition can only fail when engines are started). In order to express this, the procedures can be specified as part of a **Sequence** (all procedures have to be trained) or a **Choice** (per session only one element is trained, but all have to be trained during the complete training). More details on that are described in the section on the training manager (section 3.3).



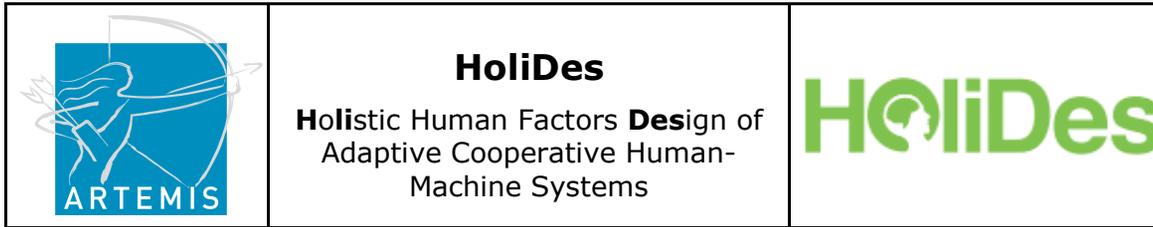
**Figure 26: Training Model**

### 3 Modelling Techniques and Tools V1.5

In this section, the editors that are based on the previously described modelling languages are explained in more detail. The following Table 5 gives an overview on the tools, and their application in the AdCoS domains. The legend for this table is the following:

<b>WP</b>	<b>AdCoS Number</b>	<b>AdCoS Description</b>
WP6	6.1	Safe patient transfer
	6.2	Guided Patient positioning
	6.3	Safe parallel transmit scanning
	6.4	Robust ECG triggering System
	6.5	iXR 3D Acquisition
	6.6	Patient data access
	6.7	Operator Task Schedule and guidance
	6.8	Querying openEHR data
	6.9	Internal analysis and reporting
WP7	7.1	Diversion Assistance
	7.2	Diversion Assistance
WP8	8.1	Command & Control Room
	8.2	Energy Control Room
WP9	9.1	Frontal Collision Scenario
	9.2	Overtaking

<b>Description</b>	<b>Code</b>
Interesting	I
Planned	P
In use	U



Tool	Extended Name / Description	Used Model	Pa rt.	WP6									WP7		WP8		WP9							
				6.1	6.2	6.3	6.4	6.5	6.6	6.7	6.8	6.9	7.1	7.2	8.1	8.2	9.1	9.2						
				PHI	PHI	UMC	PHI	PHI	ATO	IGS	ATO	ATO	HON	TRS	CAS	IRN	CRF	IAS	TAK	CRF				
MagicPED	Procedure Editor extension of MagicDraw UML Tool	Task Model	OFF		I		I	I		I				U		I	I				I			
CASCaS	Cognitive Architecture for Safety Critical Task Simulation	Cognitive Model	OFF	I	I		I			I											P			
HEE	Human Efficiency Evaluator	Task Model, Cognitive Model (CASCaS)	OFF		P		I	U	P	I	P		I		P						I			
Training Manager	Tool for creation of adapted training syllabi	Task Model, Training Model	OFF											U										
COSMODRIVE	Cognitive Simulation Model of the car Driver	Cognitive Model	IFS																		I	I	I	
GreatSPN for MDPN	Editor for MDPN used as virtual co-pilot	Petri Net Model	UTO							P			I	I							U	I	U	
BAD-MoB	Bayesian Autonomous Driver Mixture-of-Behaviors Models, Driver state inference / behaviour prediction	Human Behaviour Model	OFF																		U		U	
Driver distraction model	Model of human (audio) distraction	Human Behaviour Model	TWT										I								I	P	P	I
djnn	HMI model editor	UI / Interaction Model	ENA							I			I			I								
Driver Distraction Classifier	Driver Distraction Classifier	Human Behaviour Model	UTO																		U	I	I	U
Pilot Pattern Classifier	Pilot Pattern Classifier	Human Behaviour Model	TEC											P										

**Table 5: Methods, techniques and tools overview table**

### 3.1 MagicPED (OFF)

#### 3.1.1 Introduction

Tasks are used to describe the logical activities to reach a user's goal. A typical approach is to structure tasks (for instance by using a task hierarchy) and to relate them by temporal relations. Tasks are associated to objects that need to get manipulated to perform a task, which are in our terms Resources.

Task models allow designers to focus on the artefacts that should be realized from a user-centred point of view, instead an engineering point of view. Also, designers are forced to explicitly represent the rationale of design decisions with their task models, and different analysis of the task models allow making decisions based on objective data. For more details, see section 2.1.1.

In previous projects, OFF developed the Procedure Editor PED, which enables rapid prototyping of cognitive task models, based on a Hierarchical Task Analysis. During the initial phase of HoliDes, OFF decided to discontinue the development of the old PED and focus our efforts on a UML approach. Because integration of tools into existing tool landscapes in the industry is a tedious issue, and because OFF wanted to concentrate more on conceptual work and algorithm development, instead on editor development, it has been decided to base future development on COTS Unified Modelling Language (UML) tools. COTS UML tools can be extended via UML profiles, and often also via plugins. For OFF it seems more practicable to concentrate on development of the UML profile and plugins, because UML is widely used in the industry. Thus task modelling can become more accepted by designers, if they can stay in a tool that they are already familiar with.

A profile in the UML provides a generic extension mechanism for customizing UML models for particular domains and platforms. Extension mechanisms allow refining standard semantics in strictly additive manner, preventing them from contradicting standard semantics. Profiles are defined using stereotypes, tag definitions, and constraints which are applied to specific model elements, like Classes, Attributes, Operations, and Activities. A profile is a collection of such extensions that collectively customize UML for a particular domain.

As UML Tool, OFF has chosen MagicDraw ([www.nomagic.com](http://www.nomagic.com)) as our reference.

#### 3.1.2 State-of-the-Art

In the following sections, some task editors and their notations are described.

### 3.1.2.1 ConcurTaskTree (CTT)

The ConcurTaskTree (CTT) notation is one of the most cited approaches for tool-based task analysis and design of interactive applications. It has been designed to offer a graphical syntax that is easy to interpret and designed by using a tool, the CTT Editor. CTT can reflect the logical structure of an interactive system in a tree-like form based on a formal notation. Different to other notations like the User Action Notation [59] the CTT notation abstracts from system-related aspects to avoid a representation of implementation details. Paternò [138] states that a compact and understandable representation was one of the most important design aspects of CTT in order to enable the modelling of rather large task models for industrial applications in a compact and easy reviewable way even by people without a formal background.

A CTT tree represents a hierarchy of tasks, each categorized in one of four categories. Each layer of the tree refines the level of abstraction of the tasks until the task tree leaf nodes, that are called basic tasks, cannot be refined any further. Tasks that have the same parent task can be combined using temporal operators to indicate their relationships. An often mentioned downside of CTT is that it requires introducing artificial parental tasks to avoid ambiguities in the temporal constraints. These super tasks do not have any value for the user and make the model harder to understand. Furthermore, the support to model conditions is very limited and not explicitly available in the CTT notation and non-temporal relations between tasks are not supported (for instance the interaction that has to be entered to perform a task might depend on the data entered in a previous task). Finally the support of CTT for incremental modelling is limited as often semantics cannot be added by just editing one place but often require the addition of new tasks. CTT is the basis for the W3C task model described above in section 2.1.5.3.

The editor for CTT is called CTTE (ConcurTaskTrees Environment). Figure 27 shows a screenshot of the editor.



# HoliDes

## Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

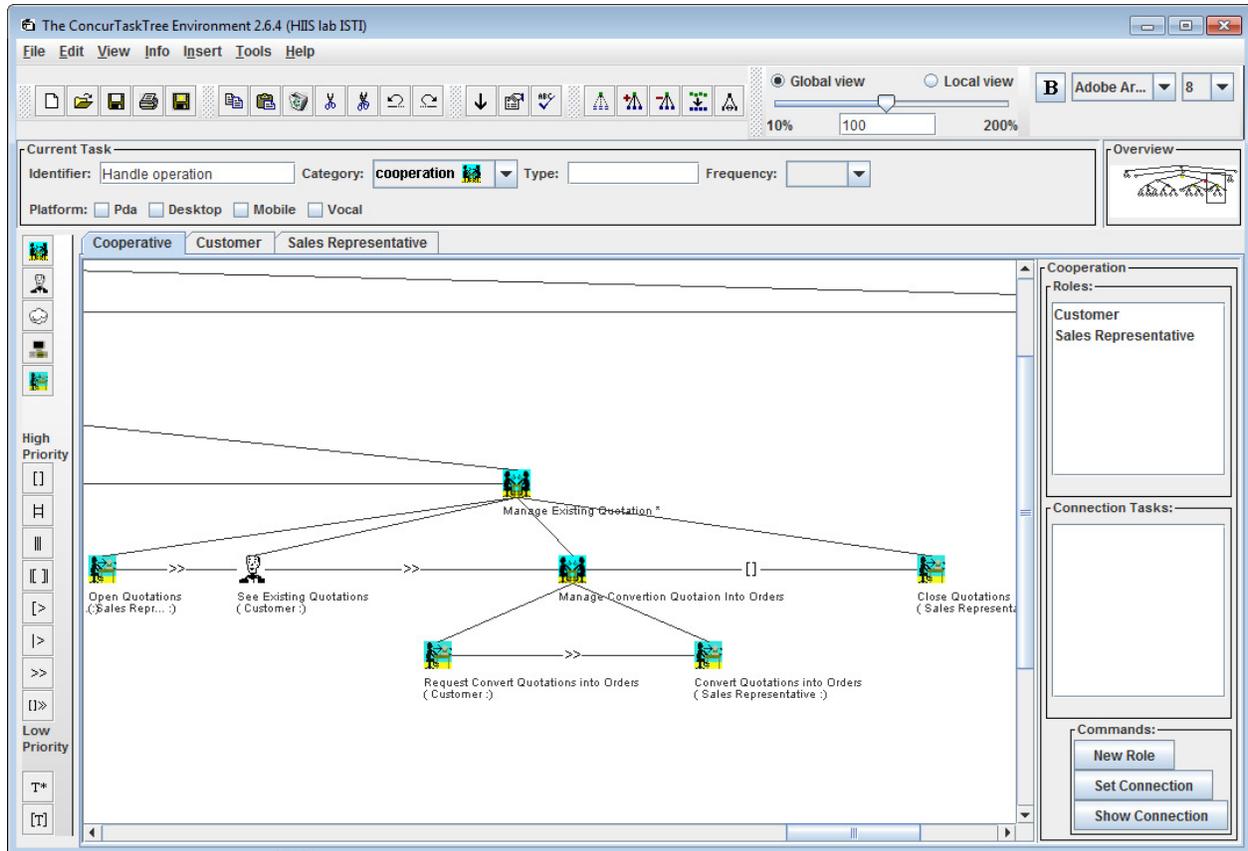


Figure 27: CTTE

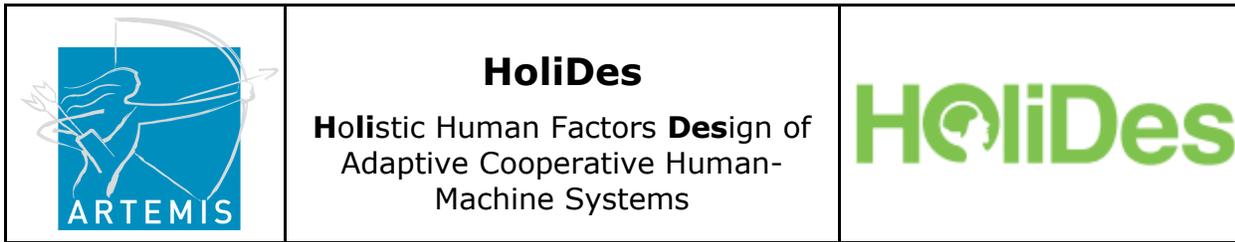
The editor allows graphical definition of a CTT, and includes validation mechanisms, as well as a simulation component.

### 3.1.2.2 GOMS

The GOMS model was developed by Card, Moran and Newell [28] as a way of quantitatively predicting the skilled and error free performance of users interacting with a text editor. Since then, GOMS has been widely extended for use with other categories of HMIs (e.g. KLM, NGOMSL, CPM-GOMS, ...).

It seems that there have been numerous editors for GOMS and GOMS derivatives developed, but most of them seem to be outdated and no longer actively maintained, e.g. the GOMSED is only available for 16 bit Windows, and does not run with actual versions of Windows.

QGOMS provides a graphical editor for "Quick and Dirty" GOMS, but it seems that it cannot be downloaded anymore.



One tool that is up-to-date and actively maintained is available is Cogulator<sup>3</sup>, a textual frontend for GOMS with a library of GOMS operators. It comes with a very nice visualisation of the calculated timing for perception, cognitive and motor components, as well as a visualisation of the memory.

### 3.1.3 MTT Description

As mentioned in the introduction OFF decided to base its task editor on a commercial UML tool: MagicDraw by NoMagic Inc<sup>4</sup>. Therefore, MagicPED consists of two parts. First the UML editor MagicDraw itself, and second a package by OFF with the UML profile for the task models and a set of plugin, extending the editor of MagicDraw.

MagicDraw provides a full featured UML editor, and provides next to model validation, an API for extending MagicDraw via plugins, also an additional TeamServer, which allows to cooperatively working on models.

In this deliverable we will use the name MagicPED for MagicDraw with the UML profile for task models and the plugins written by OFF installed.

MagicPED provides two kinds of diagrams for modelling tasks:

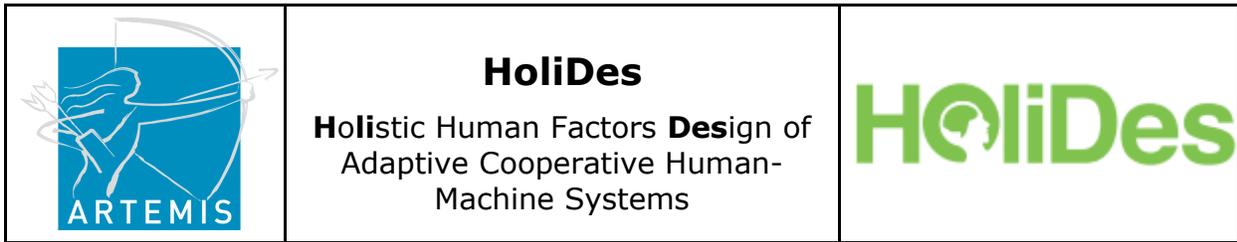
- Task Hierarchy Diagram: This diagram refers to the HTA part of the HoliDes Task Model, see D1.4 and section 2.1.6.1.
- Rule Diagram: This diagram refers to the GSM based part of the HoliDes task model, see section 2.1.6.1.

Depending on the aim of the analysis and the desired level of abstraction, only the Task Hierarchy Diagram or both diagrams are needed. In the current release (cycle 1), no analysis (metric) is provided, but in future, the following will be supported:

Analysis	Description	Needed Details/Diagrams
Simple Execution Times (GOMS like)	By expert judgement/experiments Tasks are annotated with estimated execution times. The metric calculates the execution times for higher Task levels	Task Hierarchy

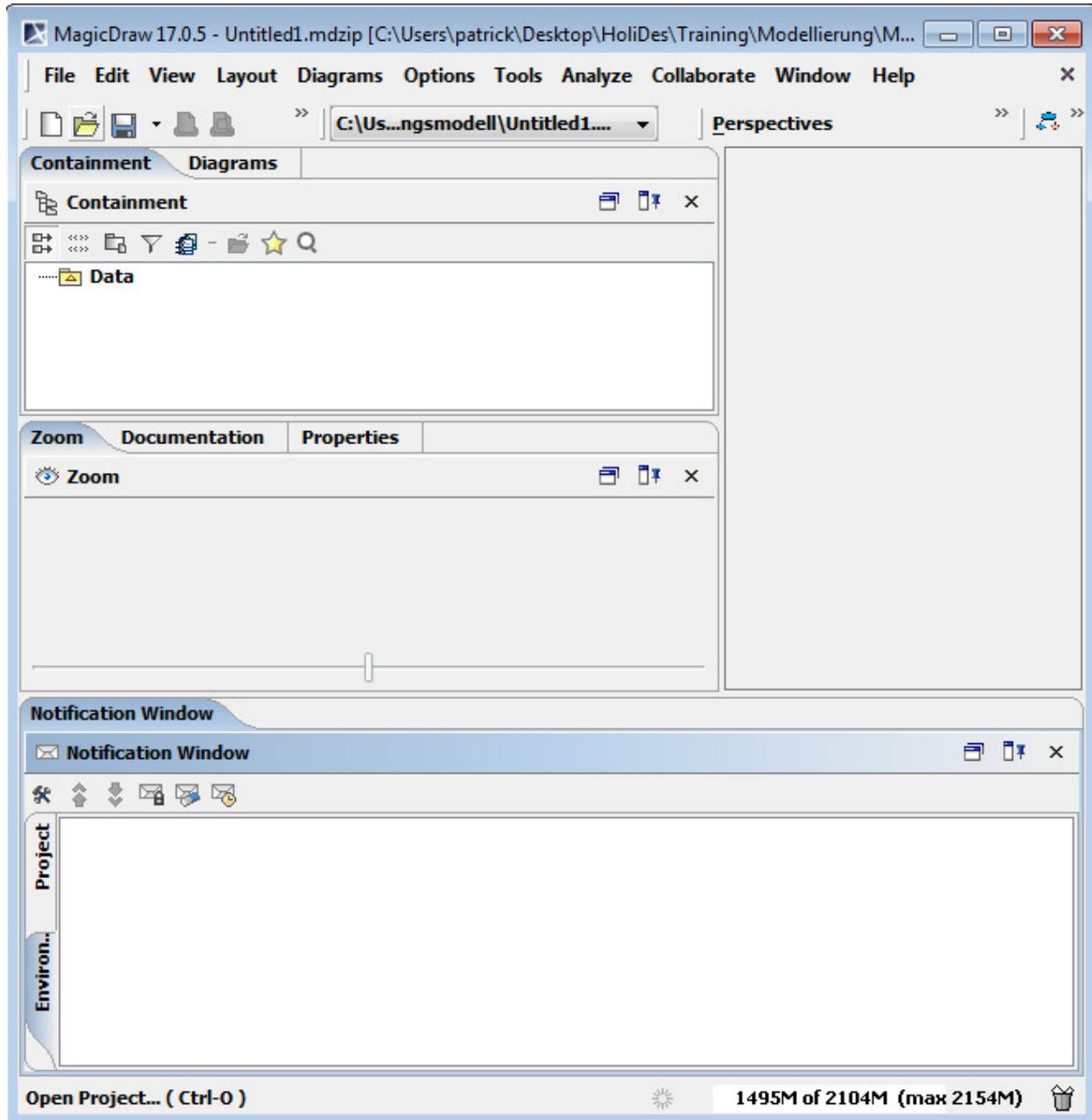
<sup>3</sup> Cogulator: <http://cogulator.io/>

<sup>4</sup> <http://www.nomagic.com/products/magicdraw.html>



Execution Time	Execution time is automatically calculated from the rules by applying a task and rule selection process similar to the one in CASCaS	Task Hierarchy with Rules
Workload	Workload calculation based on the annotated workload values in the rule elements	Task Hierarchy with Rules and annotations of workload categories at the rule elements
Simulation with CASCaS	The procedures are translated to CASCaS format and can be used there to perform a simulation.	Task Hierarchy with Rules

Figure 28 shows the main window of MagicDraw, with one project opened. On the left, the Containment Tree shows all elements in the current model. At the right, the editor component is displayed. Figure 29 shows the editor component with an opened Task Hierarchy Diagram, and Figure 30 shows an example for a Rule Diagram.



**Figure 28: MagicDraw Main Window**



# HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

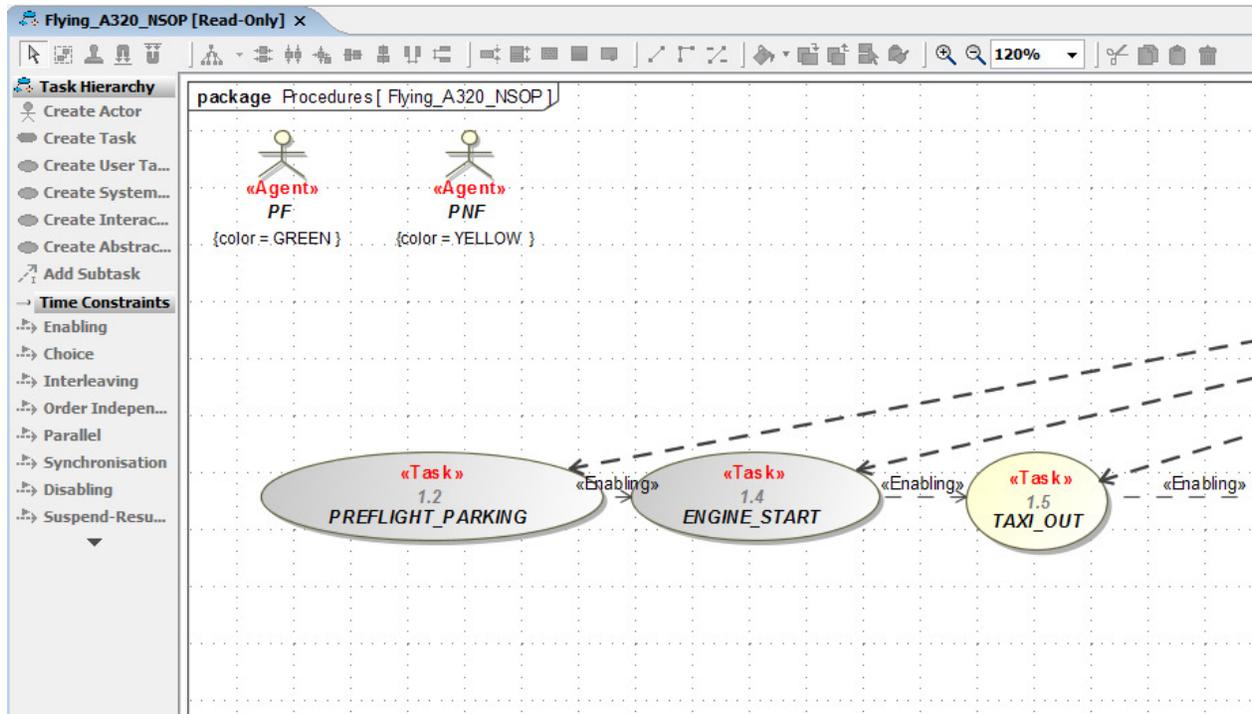


Figure 29: Editor part with Task Hierarchy Diagram

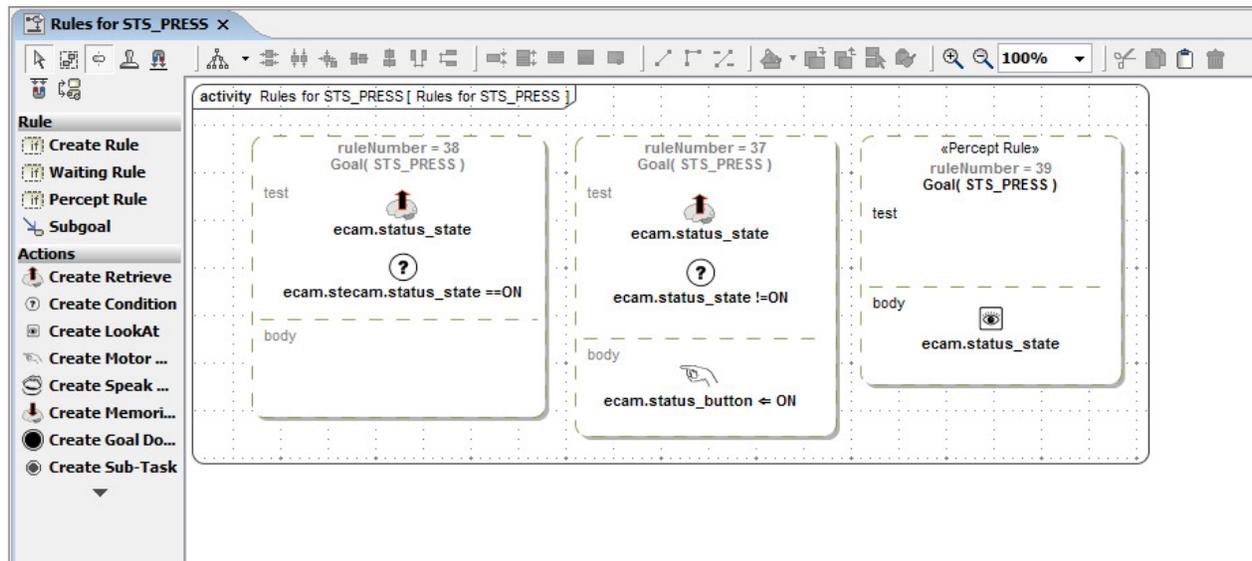
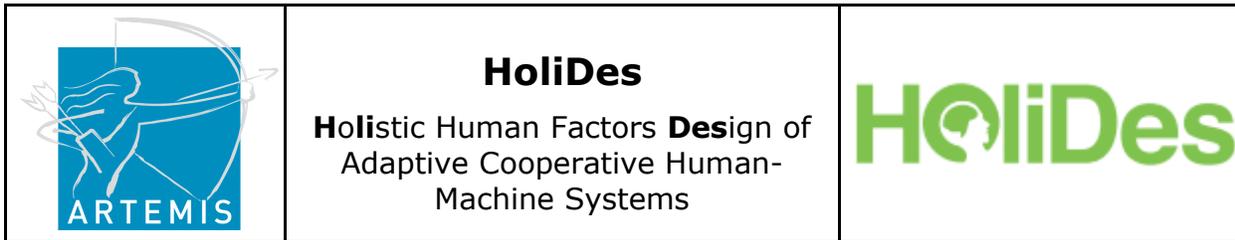


Figure 30: Rule Diagram

More details can be found in the MagicPED manual (Annex III, D2.4).



## 3.2 Human Efficiency Evaluator (OFF)

### 3.2.1 Introduction

The Cognitive Analysis of Adaptive Cooperative Systems (AdCoS) depends on complex architectures and simulations and is still driven by proprietary notations. The creation of cognitive models requires in depth cognitive modelling knowledge and is currently only accessible to experts.

New methods and techniques are therefore needed in order to ease analysis of the impact of new instruments, new display designs and their supported adaptations with respect to human factors and to make these techniques available to users without a cognitive modelling background.

Design questions that can be answered by performing a cognitive analysis with the Human Efficiency Evaluator (HEE) are:

- How does the task execution performance of the operator change with each adaptation?
- Is the workload of the operator affected?
- Does it change the average attention allocation of the operator?
- Has it an impact on the average reaction time of the operator to a specific event?

These questions are typically answered by doing tests with real users performing their task with system prototypes. User testing can result in extensive information helping to discover common errors and usability problems and in getting feedback before the final system is being implemented.

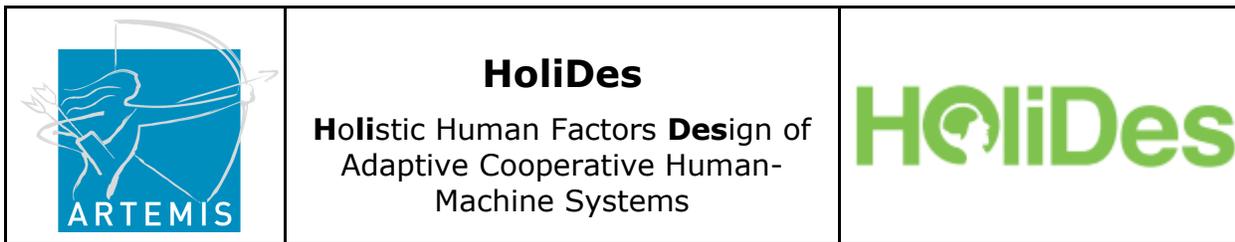
But user testing is also expensive in terms of time and money. Test users who represent the targeted audience need to be recruited and paid, which is problematic in safety-critical system domains as extensively trained operators are needed (i.e. pilots and physician).

Furthermore, user testing can only be scaled to a very limited extent: Often, because of costs and time issues, only a few variants of a design can be tested, especially, if these tests require a functional prototype to be implemented.

Usability evaluator is a modelling tool chain that consists of several tools:

- Task Editor – to identify interaction tasks between the operator and system.
- SCXML – conform State Chart Editor for instrument modelling





Offering such an early measurement gives the opportunity to consider even the most “creative” or “different” designs for an evaluation since efforts for performing such a cognitive analysis are low.

The measurement quality depends on the quality and amount of the input data. Thus, an absolute measurement (e.g. how much faster is variant X compared to Y) cannot be exactly stated in such an early phase of the design with only limited data available. Instead the focus is on comparative results (Which variant is faster?). If absolute measurements are still required, the most convincing variants can then be part of a more detailed user study, or an extended cognitive analysis.

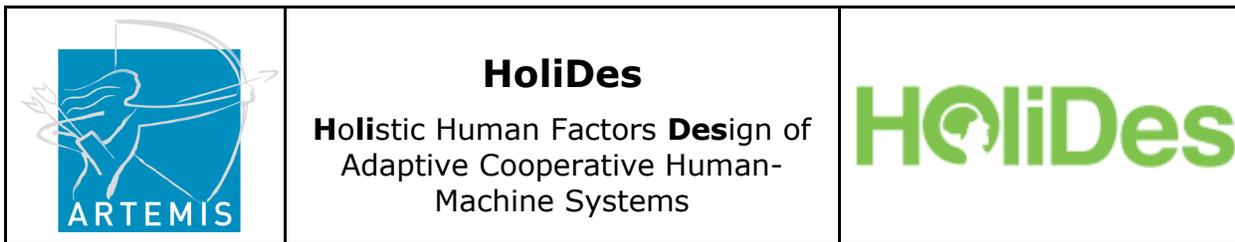
The current state of the HEE re-uses already validated models for performance prediction, such as for instance the Keystroke-level model and Fitts law. Also psychomotor actions are modelled based on validated models. Nevertheless workload predictions and attention predictions depend on partially validated models that have only been validated for other application domains.

### **3.2.2 State-of-the-Art**

The Cognitive Analysis supported by HEE is based on computational models of human cognitive processes. Cognitive models usually consist of two parts: a cognitive architecture, which integrates task independent cognitive processes (like perception, memory, decision making, learning, motor actions) and a flight procedure model which describes procedures as a temporally ordered hierarchical tree of goals (e.g. landing the aircraft), sub goals (e.g. extend flaps/slats, extend landing gear, apply air brakes) and actions (e.g. press button, move lever). Computational models are executable in a simulation environment to simulate interaction between human and machine. In order to perform such a simulation the procedure model has to be ‘uploaded’ to the architecture. Thus, a cognitive architecture can be understood as a generic interpreter that executes such formalized flight procedures in a psychological plausible way [...]

Computational cognitive models have the potential to automate parts of human factor analyses during system design. In order to leverage this potential the models have to be embedded in a design tool which can be readily applied by design experts.

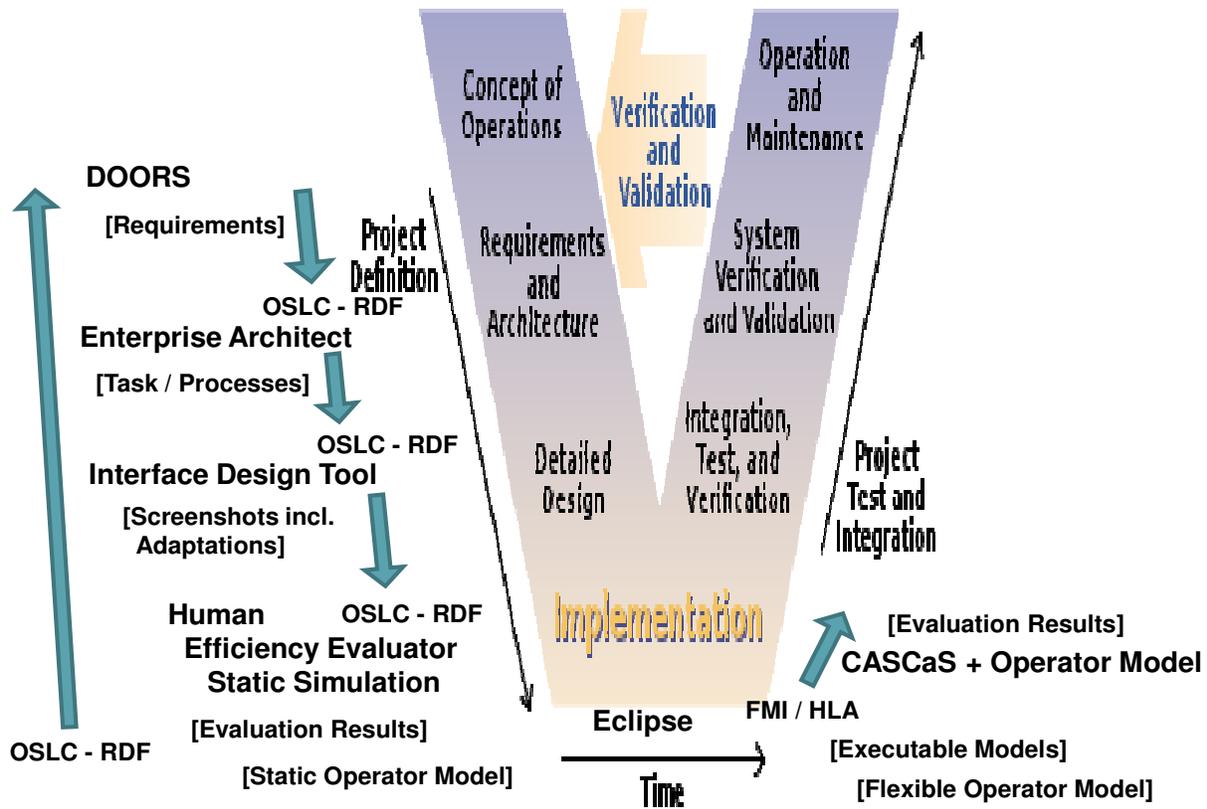
In the recent years, several tools have been proposed. The CogTool [82] enables non-experts in cognitive analysis to create predictive human performance models to estimate the task completion time for skilled human operators. CogTool is used in an early design phase. Photos or screenshots are arranged into wired frames and then annotated with interactive widgets that offer frame navigation (i.e. links or buttons), or more complex interactions, such as menu-navigation. Based on this



input, storyboards can be demonstrated by the human operator and skilled human operator performance predictions can be made and compared. CogTool is focused on predicting the task performance for interactive desktop and mobile applications (and therefore relies on a set of the most common desktop interaction widgets). Different to CogTool the HEE has been implemented to identify workload “hotspots” for human-machine interaction for safety-critical systems, such as aircrafts, control rooms and clinical healthcare systems. These systems usually rely on custom, domain specific interfaces that cannot be handled with CogTool.

The Hierarchical Task Mapper (HTAmap) framework [41] is another approach to simplify the development and analysis of cognitive models aiming to reduce the development effort. HTAmap implements a pattern-based approach: It transforms sub-goal templates gained by a preceding SGT task analysis [134] into a cognitive model by associated cognitive activity patterns (CAP). Several re-usable CAPs have been implemented to generate declarative and procedural ACT-R [7] structures e.g. to describe scanning, observation, monitoring or action execution of an operator. CAP HTAmap implements a concept for re-using concepts within a cognitive, task-centric model while the HEE implements re-usability on an instrument level by linking instrument designs to cognitive models.

The Automation Design Advisor Tool (ADAT) [154] supports comparing Flight Management System (FMS) designs in terms of their expected effects on human performance and also evaluates FMS designs based on guidelines regarding the (1) display layout, (2) how notices about changes are communicated to the pilots), the (3) meaningfulness (regarding terms and abbreviations), if the design can cause (4) confusions (e.g., similar display elements), the (5) cognitive complexity (i.e., automation surprises), and regarding their (6) procedural complexity (e.g., large number of keystrokes, requirements for unprompted actions). Designers then receive feedback on potential weakness in their proposed designs for each module based on a 1 to 10 evaluation scale. Like the HEE, ADAT is designed to be used by subject matter experts (e.g. to human factor experts in the aeronautics domain) but the conceptual foundation of both tools is different: ADAT extensively applies heuristics to evaluate the display layout, whereas the HEE relies on simulating an operator with a cognitive architecture. Both tools evaluate the design using evaluations scales. ADAT focuses on graphical design evaluation while the HEE invests in task and workload predictions.



**Figure 32: Human Efficiency Evaluator as part of a development process based on the V-Model.**

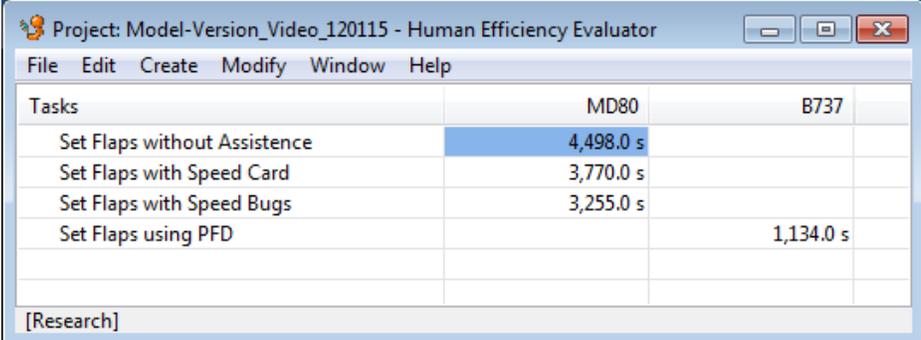
COGENT [33] is a graphical modelling editor targeted to psychologists that allows programming cognitive models at a higher level of abstraction. It is based on box/arrow diagrams that link to a set of standard types of cognitive modules that implement theoretical constructs from psychological theory. Both COGENT, CogTool, and HEE share the idea of making cognitive modelling easier by allowing programming on a higher level of abstraction. Whereas COGENT focuses on psychologists, the HEE is targeted to be used by subject matter experts.”

### 3.2.3 MTT Description

The Human Efficiency Evaluator (HEE) is a tool for cognitive analysis of design prototypes.

“It is based on CogTool because of two reasons: First, we share the idea of supporting an evaluation of interfaces based by annotating design sketches at a very early stage. Second, the results are also predictions based on a simulation.

But there are several fundamental differences between both: CogTool focuses on performance prediction of WIMP (Windows, Icons, Menus, Pointer) interfaces. Therefore, the annotation of design sketches is based on a fixed palette of the most common WIMP widgets like buttons, menu bars, and radio buttons. The annotation process to construct a user interface model is therefore straight-forward by identifying and marking widgets exactly as they are depicted in the design sketch. However, HMI have no fixed widgets (even though there are standards, e.g. colour) and new design proposals often intentionally break with some of already existing concepts.



Tasks	MD80	B737
Set Flaps without Assistance	4,498.0 s	
Set Flaps with Speed Card	3,770.0 s	
Set Flaps with Speed Bugs	3,255.0 s	
Set Flaps using PFD		1,134.0 s

[Research]

**Figure 33:** Project Overview

Therefore, we re-implemented the entire backend of CogTool to use our cognitive architecture CASCaS and integrated support for generating operator models. Furthermore, we exchanged the hard-coded widgets palette of CogTool to annotate the designs with a model-based backend that enables us to define new HMI instruments without recompiling the tool.” (Slightly adapted, taken from “Revealing Differences in Designers’ and Users’ Perspectives: A Tool-supported Process for Visual Attention Prediction for Designing HMIs for Maritime Monitoring Tasks” by Feuerstack et al., currently under review).



# HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

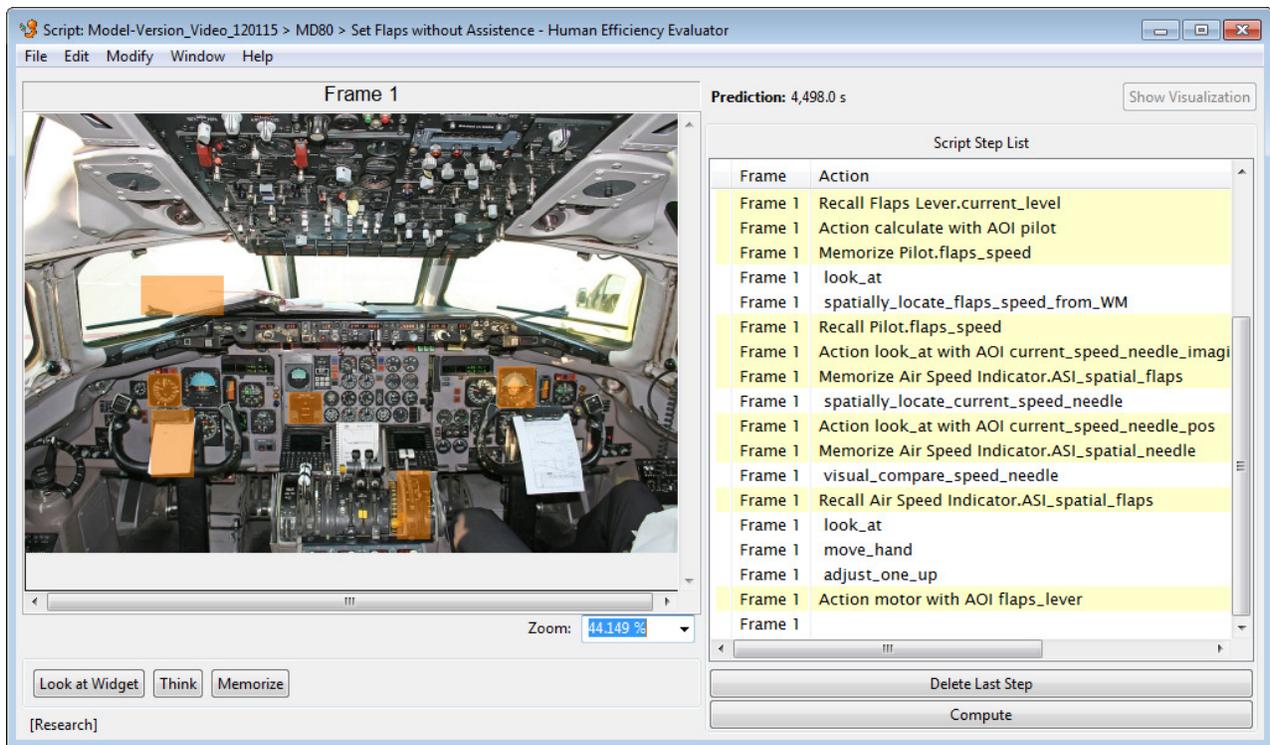


**Figure 34: HEE design annotation view.**

In the following, the first version of the HEE will be described. The conception and development of the tool started in the beginning of the HoliDes project.

After starting the HEE and selection of a pre-existing project (or after the creation of a new, empty project), the HEE displays the project overview window. The project overview lists the tasks as rows and the designs to be evaluated as columns (c.f. Figure 33). For each task/design tuple the task performance of an experienced operator can be predicted. The result of this prediction (a time in seconds) is then displayed in the corresponding cell.

Two activities have to be performed, before a task performance can be generated: First, each HMI design needs to be annotated with information defining its interaction capabilities and second, each task needs to be demonstrated for each design. The former one is supported by the design view of the HEE, which can be



**Figure 35:** HEE Procedure Demonstration.

accessed by clicking on the column name of a design. The design editor is depicted by Figure 34. A palette at the left side shows all annotation options, with a set of pre-defined widgets and instruments for the specific application domain. After the operator is “virtually” positioned relative to the design (by specifying the operator’s distance to the design and the physically correct dimensions of the design), all instruments relevant for the operator’s tasks are marked by selecting the corresponding palette entry and identifying the correct location of the instrument on the photo.

After the design has been annotated, the design annotation window can be closed and the task demonstration can be started by double-clicking on a cell that corresponds to one task/design tuple. The task demonstration window is shown by Figure 35.

The task demonstration is performed by interacting with the annotated areas of the design. Each annotated area offers instrument specific control actions reflecting different options of how an instrument can be used. Thus, a pilot might first “**look\_at**” a flaps lever to identify the current flaps setting, then “**move\_the\_hand**” to it and finally “**adjust\_one\_up**” to move the flaps lever to a new flaps level. These actions are offered context-sensitive (with respect to preceding actions), are based on finite state machine models and can be accessed by a context-menu with a right mouse-button click on an annotated area.

For each action performed, the HEE builds a demonstration script, which is depicted in the right area of Figure 35. Besides explicitly selected actions, further relevant cognitive actions are added (working memory access for instance) and are marked by a yellow background.



Such demonstration scripts correspond to a HMI operator model and can be executed by a simulation within a cognitive architecture. This functionality is embedded into the HEE and can be initiated by pressing the “compute” button ( Figure 35). The task performance time prediction is shown above the list with steps (Figure 35) and as well added to the project overview (Figure 33) to ease the comparison between different design variants.

### **3.3 TrainingManager (OFF, TRS)**

#### **3.3.1 Introduction**

The TrainingManager will be developed by OFF in cooperation with TRS within HoliDes. Objective is to develop a tool(-chain), which allows modelling of all aspects of a transition training (i.e. from one aircraft type to another), in terms of

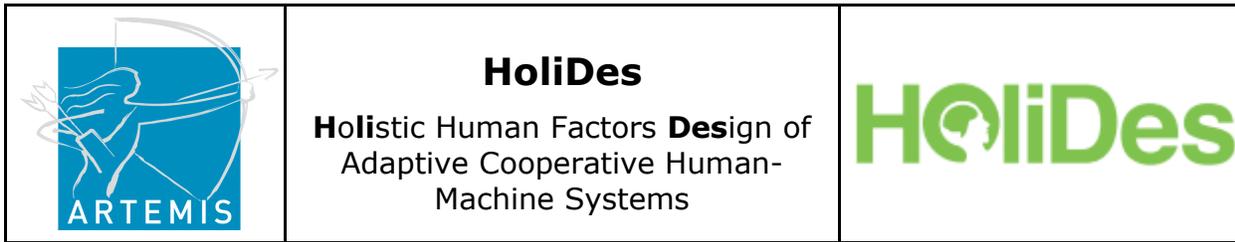
- Procedures to be trained by the trainee (SOPs)
- Flight Crew Licensing Requirement (FCLRs; coming from Regulations)
- Training syllabi, including flight phases, scenarios, ...
- Learning Knowledge

The TrainingManager will take the SOPs from two different aircrafts, and will create new training syllabi based on a scientific approach, by using the differences of the SOPs. It will also take into account latest knowledge on learning theory and practice.

The use case is to derive new training syllabi for transition from one aircraft to another, based on the SOPs and needed FCLRs. The TrainingManager will be applied in the aeronautics domain in Use Case 2 “Adaptive Flight Crew Simulator Transition Training”.

#### **3.3.2 State-of-the-Art**

There are several existing professional training management tools, like MINT[69], prodefis COURSE[70], SkyManager[71], or ETA (Education & Training Administration)[72], which are used by training organisations to manage their training. These tools allow a broad range of functionality, e.g. crew training records, electronic grading, ATQP compliance, training data analysis and many more, but are more focused on scheduling resources and record keeping of licences. To our knowledge, none of these tools allows creating adapted training syllabi, based on previous experience of the trainee.



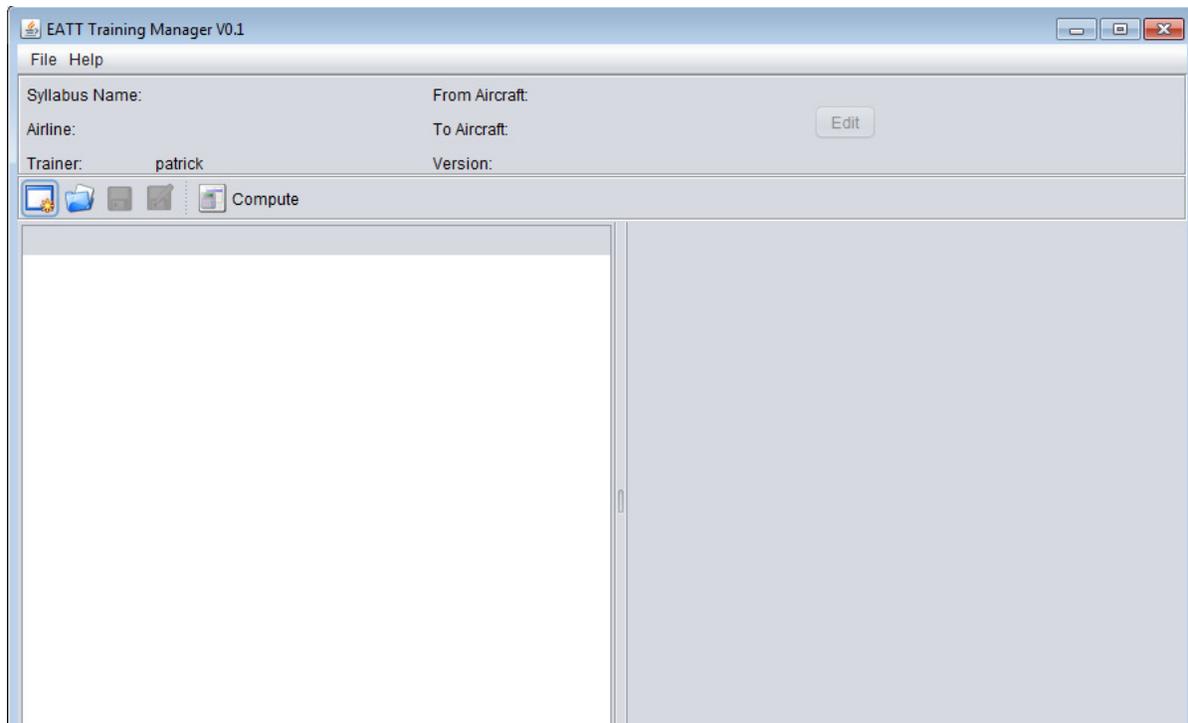
Currently, transition training is not adapted, i.e. the trainees follow the same procedure for licensing, then pilots without previous experience. Thus, the effort taken within HoliDes is completely new, and must also undergo some kind of certification by authorities at a later stage.

### **3.3.3 MTT Description**

In the following, the first version of the training manager will be described.

After logging in, the trainer sees the main window of the TrainingManager, as depicted in Figure 36. The trainer can start the creation of a new training syllabus (besides loading an existing one for editing). When creating a new syllabus, a wizard is opened, as shown in Figure 37. There he can choose from a database the Airline, as well as the procedure models (defined with MagicPED) to be used as the target and source models, which are associated with this airline (i.e. each airline has their own SOPs/aircraft, with adaptations to their flight procedures).

In addition, the trainer can specify, how many sessions are initially sold to the customer (plus check, e.g. in the figure it is eight sessions plus one check; called 8+1). The TrainingManager automatically keeps track on the versioning of the syllabus.



**Figure 36: EATT Training Manager - Main Window**

After the empty syllabus has been created (Figure 40), the trainer can start in a first step assigning content to the lessons. As explained in section 2.6 and depicted in Figure 39, the entries of a session are structured into building blocks, e.g. for the task of cockpit preparation it makes no sense to train only one of the two procedures (PRELIMINARY\_COCK-PIT\_PREPARATION, COCKPIT\_PREPARATION), and of course only one malfunction per engine start will be trained in one engine starting procedure (therefore the malfunctions are structured in a choice element).

The TrainingManager will conduct several checks, that the resulting assignments are consistent and feasible. A collection of these checks is currently on-going.



Syllabus Properties

Syllabus Name: TT\_ABC\_A320\_B737

Airline: ABC

Target Aircraft Type: A320

Source Aircraft Type: B737F

# Lessons: 8

Lesson Nr.	Trainee 1	Trainee 2	Check
1	1	1	<input type="checkbox"/>
2	2	2	<input type="checkbox"/>
3	3	4	<input type="checkbox"/>
4	4	3	<input type="checkbox"/>
5	5	6	<input type="checkbox"/>
6	6	5	<input type="checkbox"/>
7	7	8	<input type="checkbox"/>
8	8	7	<input type="checkbox"/>
9	9	9	<input checked="" type="checkbox"/>

Version	Author	Description
V0.1	patrick	Initial Version

OK Cancel

Figure 37: EATT TrainingManager - Syllabus Wizard





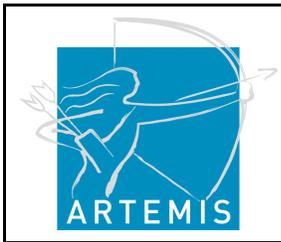
# HoliDes

**H**olistic Human Factors **Design** of  
Adaptive Cooperative Human-  
Machine Systems



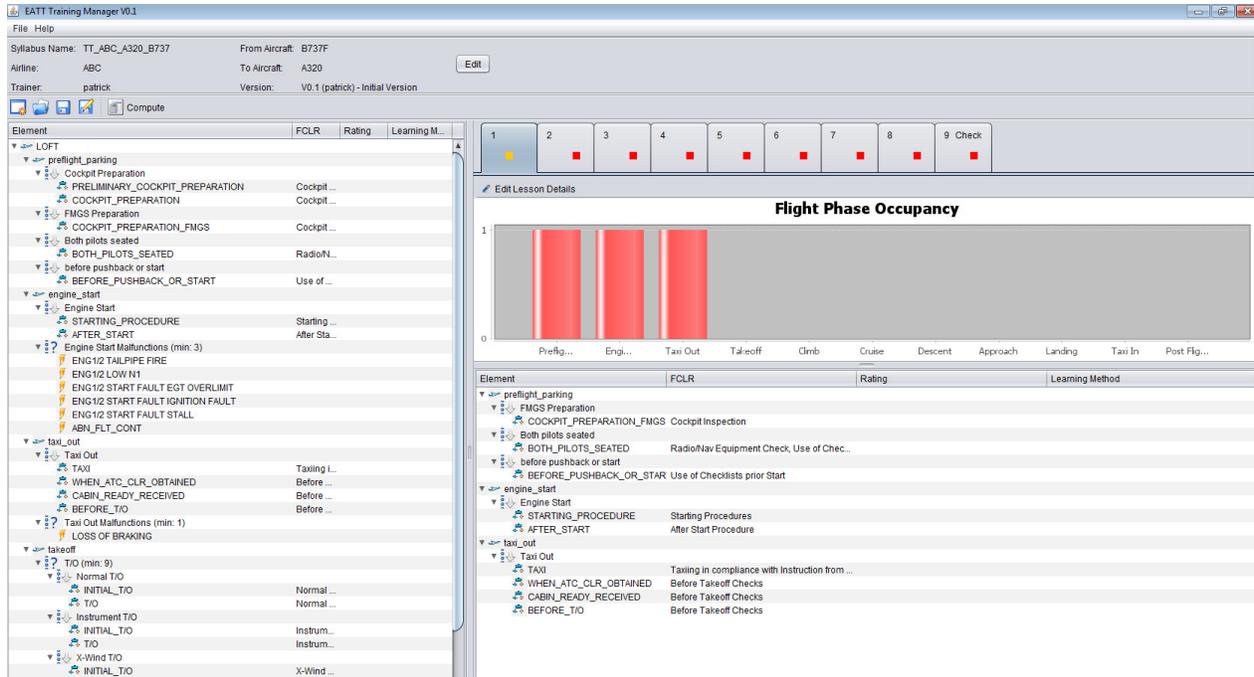
Element	FCLR	Rating	Learning Method
LOFT			
preflight_parking			
Cockpit Preparation			
PRELIMINARY_COCKPIT_PREPARATION	Cockpit In...		
COCKPIT_PREPARATION	Cockpit In...		
FMGS Preparation			
COCKPIT_PREPARATION_FMGS	Cockpit In...		
Both pilots seated			
BOTH_PILOTS_SEATED	Radio/Nav...		
before pushback or start			
BEFORE_PUSHBACK_OR_START	Use of Ch...		
engine_start			
Engine Start			
STARTING_PROCEDURE	Starting Pr...		
AFTER_START	After Start ...		
Engine Start Malfunctions (min: 3)			
ENG1/2 TAILPIPE FIRE			
ENG1/2 LOW N1			
ENG1/2 START FAULT EGT OVERLIMIT			
ENG1/2 START FAULT IGNITION FAULT			
ENG1/2 START FAULT STALL			
ABN_FLT_CONT			
taxi_out			
Taxi Out			
TAXI	Taxiing in ...		
WHEN_ATC_CLR_OBTAINED	Before Ta...		
CABIN_READY_RECEIVED	Before Ta...		
BEFORE_T/O	Before Ta...		
Taxi Out Malfunctions (min: 1)			
LOSS OF BRAKING			
takeoff			
T/O (min: 9)			
Normal T/O			
INITIAL_T/O	Normal T...		
T/O	Normal T...		
Instrument T/O			
INITIAL_T/O	Instrumen...		
T/O	Instrumen...		

**Figure 39: Tree with Lesson Entries**



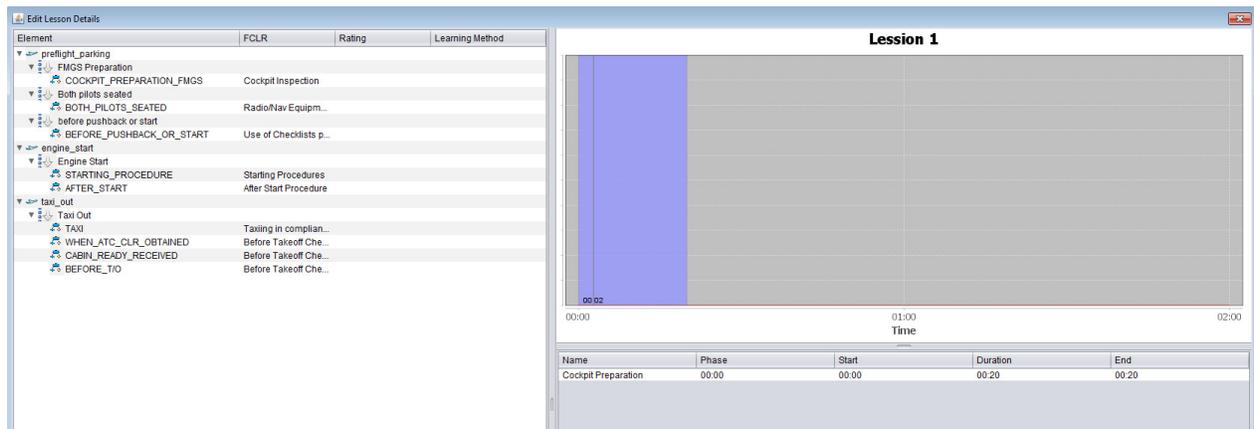
# HoliDes

## Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

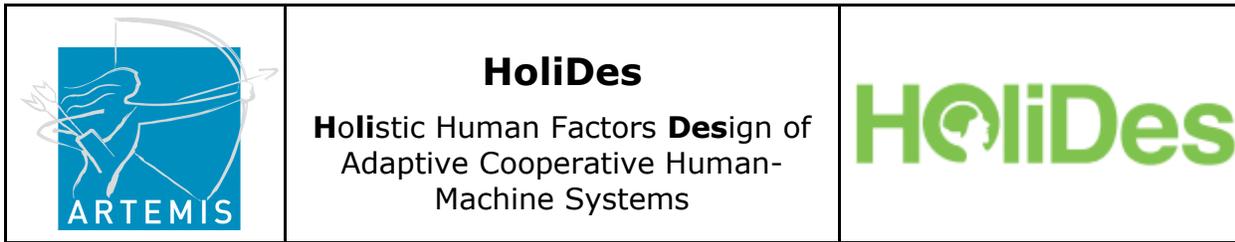


**Figure 40: EATT TrainingManager - Step 1 with assigned FCLRs**

After the requirements (flight crew licensing requirements; FCLR) have been allocated to the sessions (see Figure 40), the trainer can start the second step, the fine planning of each session, see also Figure 41, where all content that has been assigned to the session is assigned on a timeline. Also here, consistency checks are applied.



**Figure 41: EATT TrainingManager - Step 2**



## 3.4 Djnn (ENA)

### 3.4.1 Introduction

djnn is a programming framework that relies on a model of interactive software in which any program can be described as a tree of interactive components. The execution of a program is described by the interactions between its components, and between them and the external environment: components react to events detected in their environment, and may themselves trigger events.

For instance, a simple “hello world” program can be described with two components. The first prints text when activated and the second binds the activation of the first to the start of the program. Launching the program is an external event that triggers the start of the program, thus triggering the binding component which itself triggers the text-printing component.

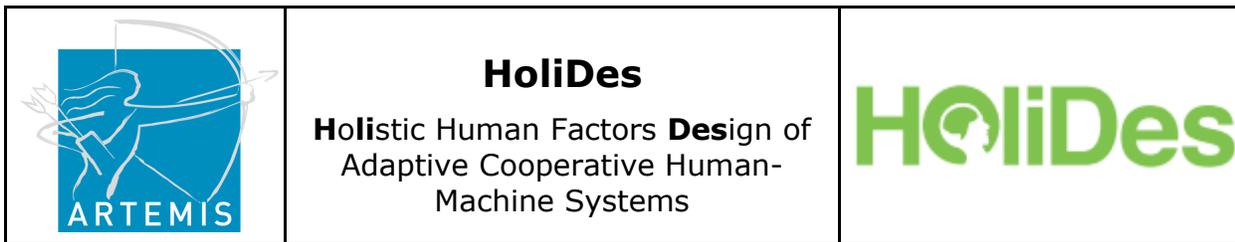
djnn has the expressive potential of a general programming language. This contrasts with most user interface programming frameworks, which provide reusable components and architecture patterns that programmers combine with code written in a traditional programming language. Not only does djnn aim at covering 100% of the user interface code, it also has the potential of describing the functional core as well, thus covering whole interactive applications.

Creating a program as a hierarchy of components is similar to constructing an abstract syntax tree, as done by traditional compilers. The tree contains all the information needed to execute the program directly or translate it into executable code for a given platform.

### 3.4.2 State-of-the-Art

For more than 30 years, dedicated languages and methods have been designed and used to deal with the development of critical systems (transportation, health, nuclear, military systems). These languages and methods are used for the development of safe, functionally correct systems. For example VHDL is hugely used for the development of hardware circuit, or SCADA (Supervisory Control And Data Acquisition) is used for control and command systems.

However, highly interactive and adaptive systems have recently and progressively appeared. For example, air traffic control systems, surveillance systems or automotive systems have to react to many event sources: user events (from classic keyboard/mouse to more advanced interaction means such as multi-touch surfaces, gesture recognition and eye gaze), pervasive sensors, input from other subsystems, etc.

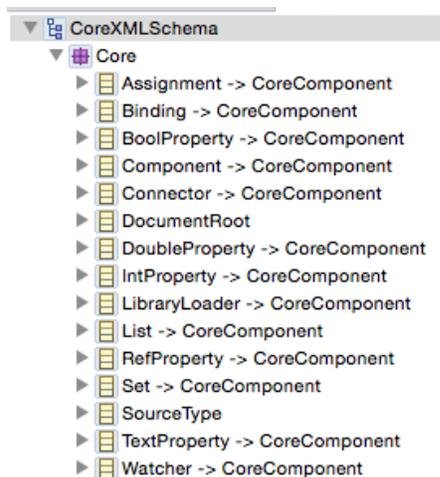


Difficulties have been observed in using existing languages and methods on these kinds of systems, due to either the weakness of their expressive power or due to the great heterogeneity of their constructs. Indeed, these systems require new control structures in order to manage dynamicity or to support different design styles, such as state machines and data flows. Part of these issues are due to the lack of a well-defined language for representing and describing interactive software design in a way that allows, on the one hand, system designers to iterate on their designs before injecting them in a development process and on the other hand, system developers to check their software against the chosen design. The djnn framework intends to provide an innovative way to address these difficulties.

### 3.4.3 MTT Description

The djnn programming framework is organized around several modules. We present here three of them which contain the main building blocks for programming a graphical user interface.

#### 3.4.3.1 The core module



**Figure 42: Components of the core module**

The core module encompasses the basic mechanisms and the basic ontology of djnn. Within djnn, everything is an element. This can be thought as the basic class in a class hierarchy. Then, some elements can hold other elements, in the XML idiom they are called component. The definition of a new component "colour1", for example, will look like this:

```
<core:component name="colour1">
```



## HoliDes

Holistic Human Factors **Design** of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

```
<core:int-property name="r" value="0"/>  
<core:int-property name="g" value="0"/>  
<core:int-property name="b" value="0"/>  
</core:component>
```

The core module also contains some basic control structures: connector and binding. A connector is a data flow component that copies the values of a property into another each time the first one is changing. For example if I want to connect the 'r' property of two colours, I can do it with this code:

```
<core:connector name="c1" in="colour1/r" out="colour2/r"/>
```

The binding propagates the control flow from a source to an action each time the source is activated. For example, if I want to emit a beep each time the 'r' property of "colour2" is changed I can write:

```
<sound:beep name="myBeep"/>  
<core:binding name="b1" src="colour2/r" action="myBeep"/>
```

### 3.4.3.2 The base module

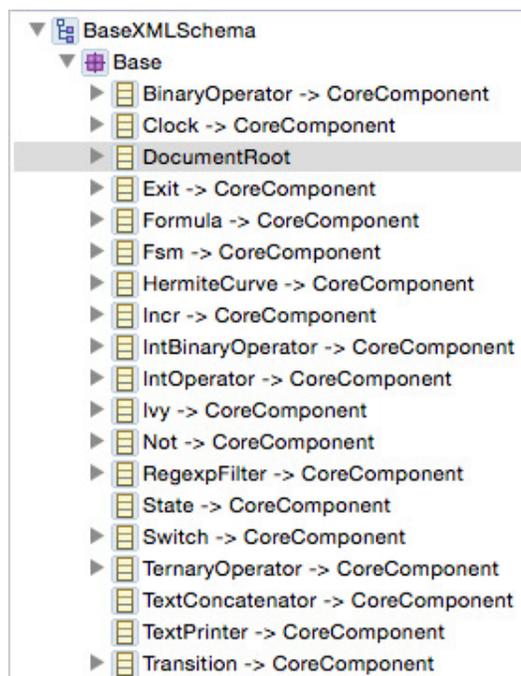


Figure 43 Components of the base module



## HoliDes

Holistic Human Factors Design of  
Adaptive Cooperative Human-  
Machine Systems

# HoliDes

The base module contains more elaborated control structures, mathematical components and few other utilities. The adder component for example is a binary operator that adds two input properties and propagates the result each time one of the inputs has changed.

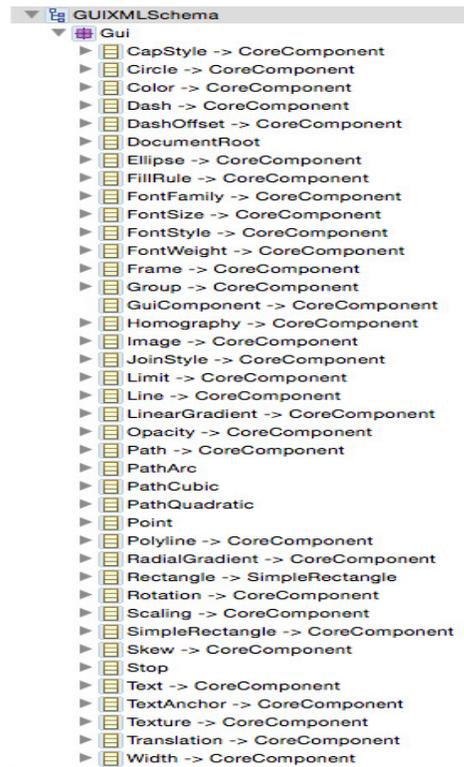
```
<core:double-property name="x" />
<core:double-property name="y" />
<core:double-property name="xy" />
<base:adder name="add" left="0" right="0" />
<core:connector name="c1" in="x" out="add/left" />
<core:connector name="c2" in="y" out="add/right" />
<core:connector name="c3" in="adder/result" out="xy" />
```

An important control structure in the base module is the finite state machine (FSM) and the possibility to combine it with a Switch. A Switch is a component that can hold several branches and which warrants that only one is active at a time. By connecting a FSM to a Switch it is possible to control which branch is active and to pass from one branch to another when some event is triggered. A skeleton for a two states component whose state change every 500ms could look like this:

```
<core:component name="myComponent" />
  <base:clock name="cl" period="500" />
<base:switch name="sw" initial="idle">
  <core:component name="idle" />
  <core:component name="pressed" />
</base:switch/>
  <base:fsm name="fsm">
    <base:state name="idle" />
    <base:state name="pressed" />

    <base:transition name="t1" from="idle" to="pressed" src="cl/tick" />
    <base:transition name="t2" from="pressed" to="idle" src="cl/tick" />
  </base:fsm>
  <core:connector name="c1" in="fsm/state" out="sw/state" />
</core:component>
```

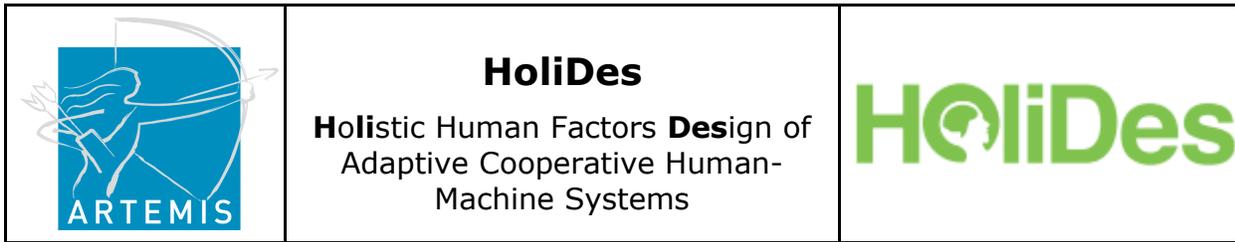
### 3.4.3.3 The GUI module



**Figure 44 Components of the GUI module**

The GUI module holds the graphical components of djnn, which are basic shapes, styles and geometrical transformations. It is important to note here that, contrarily to several graphical toolkits, graphical shapes are not the children of a window. Given a tree of component, a graphical shape is drawn in the first window on its left. It is also important to note the graphical style of a shape (stroke and fill) is not included in it. The style of a shape is fixed by the style component on its left. For example, the following xml code will produce a blue rectangle.

```
<gui:frame name="f" title="myFrame" x="50" y="50" width="500"
height="300"/>
<gui:fill-color name="fc" r="0" g="0" b="255"/>
<gui:rectangle name="r" x="50" y="50" width="100" height="50" rx="5"
ry="5"/>
```



## **3.5 Human Operator Models**

### **3.5.1 CASCaS (OFF)**

#### **3.5.1.1 Introduction**

The Cognitive Architecture for Safety Critical Task Simulation (CASCaS) is a framework for modelling and simulation of human behaviour. Its purpose is to model and simulate human machine interaction in safety-critical domains like aerospace or automotive, but in general it is not limited to those specific domains.

#### **3.5.1.2 State-of-the-Art**

Today, human error is one of the main factors in transportation accidents. In aeronautics 60-80% of commercial aircraft accidents [21] and in automotive 84% of car accidents [166] are caused by human errors. In order to further reduce the accident rates, more and more automation is introduced in cars and airplanes. Increasing automation, and the role change of the operator from active control to supervisory control, introduces new risks for human errors (e.g. [153]). New methods and techniques are therefore needed in order to analyse the impact of those systems with respect to human factors. Typical design questions like "How do the tasks of the driver/pilot change with the new system", "Does the system improve the situation awareness", or "How does the situation awareness of the driver/pilot changes" have to be answered. Current industrial practice is building physical mock-ups with prototypes, and to test the system with human subjects (test drivers or test pilots). This approach is very expensive and time consuming, thus methods and tools are needed that are applicable in early design phases, e.g. a model-based approach for Human Machine Interface (HMI) evaluation. A model-based HMI evaluation can be used for evaluation of different system designs and the induced behavioural adaption of the user. Main idea in this approach is to perform in an early design phase a computer simulation of the models/prototypes, including a model for the human behaviour (cognitive modelling as a method). This can be seen as the natural extension of the digital prototyping, where simple mock-ups and prototypes of a new system can be analysed [80], [15].

Cognitive modelling aims at creating models of cognitive processes of individual human agents. A common approach is to define a cognitive model as a set of production rules, which implement human behavioural procedures, enabling it to react on changes and manipulate states in its environment.

Among the prime benefits of cognitive modelling are executable models which capture the behaviour of a human agent interacting with a simulation environment. For instance, cognitive models hereby allow risk assessment by prediction of human performance in simulated, potentially hazardous situations.



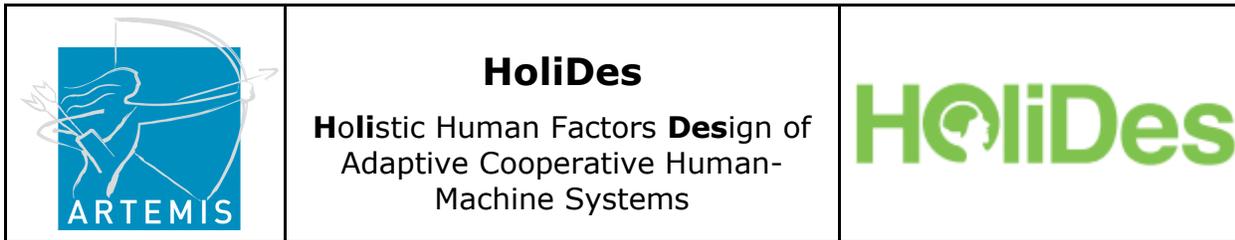
Cognitive Architectures are tools, which provide executable models of human behaviour, based on psychological and physiological models of human behaviour. At OFF, the cognitive architecture CASCaS (Cognitive Architecture for Safety Critical Task Simulation) has been developed since 2004 [117]. Main driver for the development of CASCaS is the more industrial oriented approach, and the objective to support real- and fast time simulation of human behaviour. In contrast to that, most cognitive architectures are developed for creation and evaluation of theories and models of human cognition. The best known cognitive architectures are ACT-R (Adaptive Control of Thought – Rational, [7], [8]), SOAR [130], [103] and MIDAS [34], [50], [55]. These architectures have been applied in the past to predict pilot or driver behaviour. For example, the Human Performance modelling (HPM) element within the System-Wide Accident Prevention Project of the NASA Aviation Safety Program performed a comparison of error prediction capabilities of five cognitive architectures [49], including ACT-R and (Air-) MIDAS. CASCaS has been applied in several projects, in order to model perception [115], attention allocation [180], decision making of drivers [170] and human errors of aircraft pilots [114] and car drivers [116].

Within HoliDes, we will extend CASCaS with the calculation of a Saliency Map, which helps to answer the question if certain information is salient enough to be recognised during the course of actions. It can also be used for implementing an unguided search in an environment, e.g. where does the driver look while looking through the front window. A lot of work in this direction has been done by Itty and Koch, e.g. [76], [75], [77]. This extension is currently under development.

### **3.5.1.3 MTT Description**

#### **3.5.1.3.1 Input**

The introduction gives a very rough overview of how the architecture simulates human behaviour. Two specific input files are required for a simulation: a procedure and a variable specification file. Both files are loaded into the architecture at start-up. The procedure file specifies task and domain specific knowledge about how the model should interact with its environment, for example: "If display X shows value Y I have to press button Z". The architecture itself is domain and task independent: only by loading an appropriate procedure file it becomes, for example, a driver model or a pilot model. For both of those domains (automotive / aeronautic) OFF has developed models which can interact in specific scenarios and driving / flight simulator environments, e.g. a driver model which can drive on a two lane German Autobahn, performing free-flow, car-following and lane change manoeuvres, or a pilot model which can simulate the cockpit interaction necessary by a pilot for take-



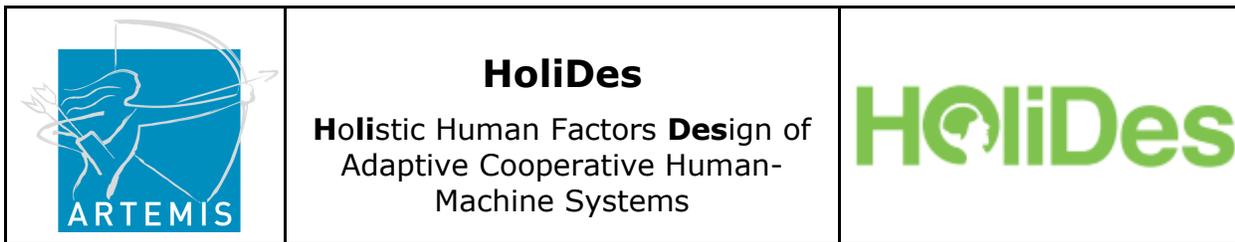
off or approach scenarios. The second file (the variable specification file) is used to define the data which is exchanged between one or more simulators and CASCaS, as well as the topology of the environment (i.e. where certain information is located in space).

Integration of CASCaS into simulation environments can be done either by point to point connections using UDP or TCP/IP sockets or by integrating all components into an HLA simulation platform. For the latter one OFF uses the open source CERTI High-Level Architecture (HLA) Implementation and has implemented several different HLA federates. CoSimECS, supports setting up the simulation by allowing graphical configuration of the HLA simulation.

#### 3.5.1.3.2 Use Cases

In the aeronautics domain CASCaS was already used to simulate procedural tasks for specific flight manoeuvres. The task execution times as well as the gaze behaviour are outputs which can be used for statistical analysis purposes. Alternative task procedure designs can be simulated and compared against each other. Designing a new cockpit system always requires the specification of operating procedures. With such a simulation, engineers can check if a new procedure for a system covers the necessary functions in certain test scenarios. At a very first pure software simulation stage it allows first feedback about possible interaction problems, e.g. if necessary information is cluttered, the task execution time will automatically raise.

In the automotive domain OFF has developed a driver model which is already able to deal with the intended scenario of WP9. The model can simulate free-flow, car-following and lane changes right and left on a two-lane German Autobahn with medium traffic density. The model simulates gaze behaviour including a mirror view which covers blind spot problems. The model heavily relies on peripheral and foveal vision which is necessary to interact with the highly dynamic traffic environment. Intention based top-down behaviour (model wants to do a lane change) as well as reactive bottom-up behaviour (model detects that a car has set its indicator) are important parts of the model. The existing driver model can be used in HoliDes and it can be extended by additional operational procedures to interact with an Adaptive Driver Assistant System (ADAS). The output is similar as for the aeronautic domain. Task interaction time and gaze behaviour can be analysed. Additionally, the impact of secondary tasks on, for example, distance keeping and lane changing could be analysed.



### 3.5.1.3.3 AdCoS Use-Cases

In the domain of driver modelling OFF and TAK have agreed on a use case where the model should be used to simulate the interaction between the driver and the HMI developed by TAK. The targeted Use-cases are WP9 TAK1-5.

In general, CASCaS is integrated into the Human Efficiency Evaluator (HEE), thus all applications of HEE include CASCaS.

### 3.5.1.3.4 Output

Output of a CASCaS simulation run is a CSV-File with the following recordings in a 50ms interval:

- Environment variables received from or send to the simulation via HLA,
- Selected goal and rule
- Goal agenda
- Actions of motor components (hands, voice)
- Actual gaze position

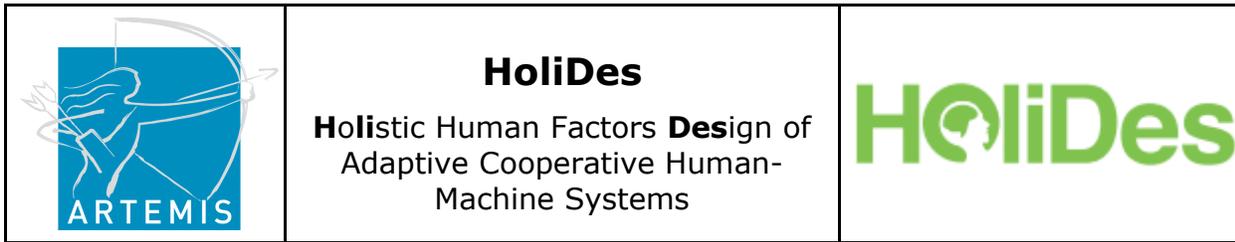
This can be used in analysis software to assess previously defined metrics (gaze behaviour, reaction times, task execution times ...). The output will be enhanced with the implementation of the saliency map, e.g. the most salient area of interest could be locked two, beside the saliency map itself.

## 3.5.2 COSMODRIVE and COSMO-SIVIC (IFS)

### 3.5.2.1 Introduction

COSMODRIVE is a Cognitive Simulation Model of the car Driver developed at IFS, in order to provide computational simulation of car drivers. The general objective is to virtually simulate the human drivers' perceptive and cognitive activities implemented when driving a car, through an iterative "Perception-Cognition-Action" regulation loop. Through this main regulation loop, the model allows to:

- Simulate human drivers perceptive functions, in order to visually explore the road environment (i.e. *perceptive cycle* based on specific driving knowledge called "schemas"; [16]) and then to process and integrate the collected visual pieces of information in the Cognition Module.
- Simulate two core cognitive functions that are (i) the elaboration of *mental representations* of the driving situation (corresponding to the driver's Situational Awareness; [17]) and (ii) a *decision-making* processes (based on these mental models of the driving situation, and on an *anticipation process* supported by dynamic mental simulations)
- Implement the driving behaviours decided and planned at the cognitive level, through a set of effective actions on vehicle commands (like pedals or



steering wheels), in order to dynamically progress along a driving path into the road environment.

### **3.5.2.2 State-of-the-Art**

COSMODRIVE is a long term research programme of IFS dedicated to cognitive simulation of human drivers [16], [17], [18]. Several preceding versions of this model (from initial theoretical framework started in 1998 to last computational simulation models implemented during ISI-PADAS project) have already existed before HoliDes project. However, a totally new version of the model has been designed and developed specifically for this project, in order to be used in WP4 and in WP9 for Virtual Human Centred Design (V-HCD) of future AdCoS. In the specific "HF-RTP" approach of HoliDes, COSMODRIVE plays the role of one of the "Human Factor" (HF) models (focused on car driving), interacting with a virtual RT-Platform (here based on RT-Maps and Pro-SIVIC tool chain, that are MTT proposed in HoliDes by 2 other partners: INT and CIV).

From the interfacing of COSMODRIVE, RT-Maps and Pro-SiVIC in HoliDes, it is possible to have a Virtual HCD platform for supporting AdCoS design and test, able to generate dynamic simulations of a driver model (COSMODRIVE), interacting with a virtual road environment (simulated with Pro-SIVIC), through actions on a virtual car (simulated with Pro-SIVIC), equipped of Virtual AdCoS (based on ADAS models and driver Monitoring Functions developed by IFS, interfaced with RT-Maps and Pro-SIVIC). This tool chain was designed during the first year of the project, and is now under the final development step. The next step (2<sup>nd</sup> Milestone) will be to use it in WP4 and WP9 for virtual design and evaluation of AdCoS for Automotive domain (as illustrated in the next figure).

### **3.5.2.3 MTT Description**

The functional architecture of the new version of the COSMODRIVE model specifically implemented for HoliDes is composed of three main modules (Figure 45): a Perception Module (in charge to simulate human perceptive information processing), a Cognition Module (in charge to simulate driver's situation awareness, anticipation and decision-making processes), and an Action Module (in charge to simulate executive functions and vehicle control abilities) generating an effective driving performance.



# HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

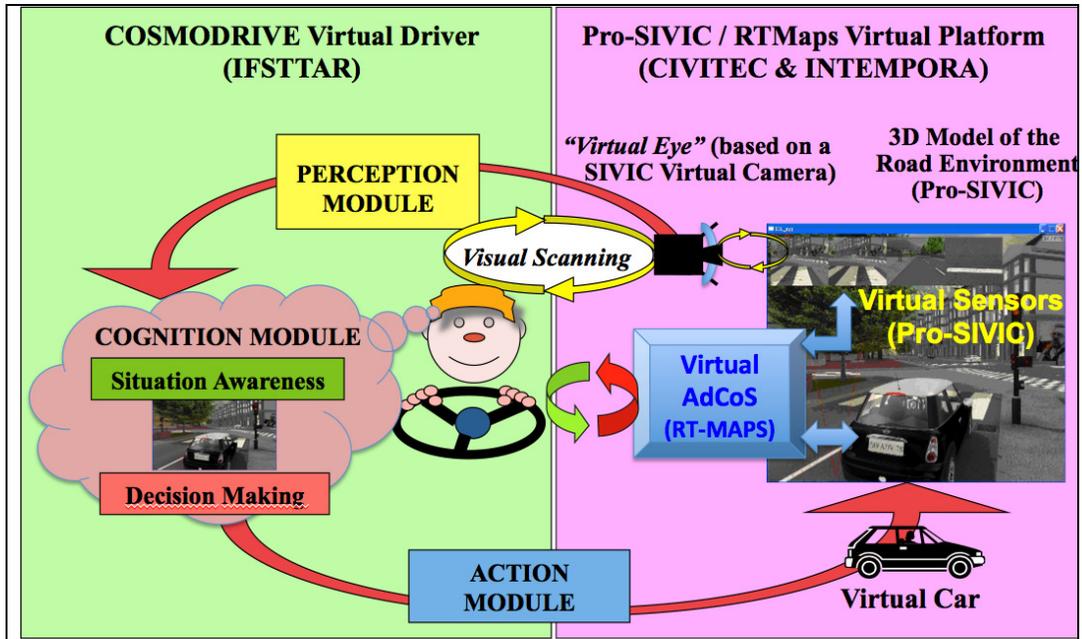


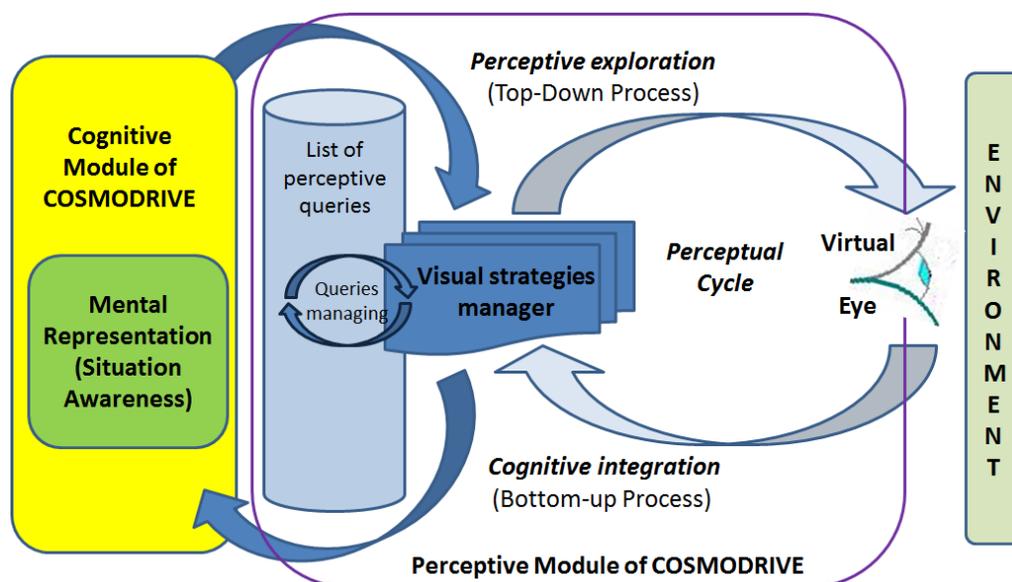
Figure 45: COSMODRIVE use in HoliDes for AdCoS and car driving simulation (supported by Pro-SIVIC and RT-MAPS software)

Moreover, the aim of the use of COSMODRIVE model in HoliDes is not only to simulate these perceptive, cognitive and executive functions in an optimal way, but also to simulate some drivers errors in terms of misperception of event, erroneous situational awareness, or inadequate behavioural performance, due to visual distractions (resulting of a secondary task to be performed during driving, for instance).

In this context, one of the core components of COSMODRIVE for HoliDes objectives is the Perception module. Indeed, the AdCoS to be designed and developed by IFS in WP3 will be in charge to monitor drivers' visual scanning (as simulated from COSMODRIVE or observed among Real Human). At last, this AdCoS based on MOVIDA functions (for Monitoring of Visual Distraction and risks Assessment) will be an integrative co-piloting system supervising a set of simulated Advanced Driving Aid Systems (ADAS), to be centrally managed in an Adaptive and Cooperative way by MOVIDA module, according to the drivers' visual distraction states and to the situational risks assessment. According to these MOVIDA-AdCoS design objectives, realistic simulation of drivers' visual scanning via the Perception Module is of prior importance.

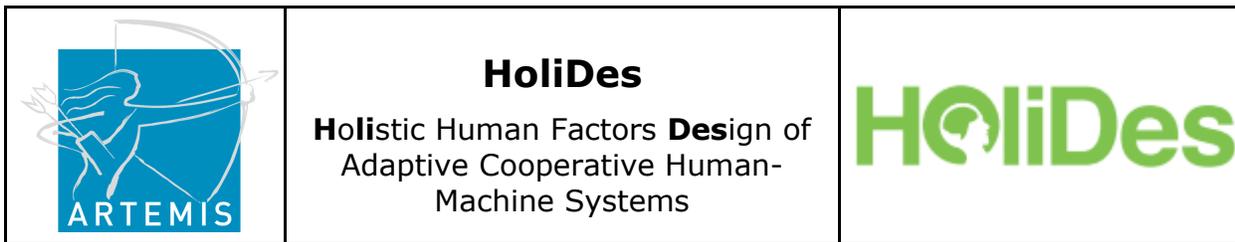
First of all (Figure 46), the COSMODRIVE Perception module integrates a Virtual Eye. This virtual eye includes three visual field zones: the central zone

corresponding to foveal vision (solid angle of 2.51 centred on the fixation point) with a high visual acuity, para-foveal vision (from 2.51 to 91), and peripheral vision (from 91 to 1501), allowing only the perception of dynamic events. Moreover, two complementary perceptive processes are implemented in this module, in order to simulate the human information processing while driving. The first one, perceptive integration, is a “data-driven” process (i.e. bottom-up integration based on a set of perceptive filters) and allows cognitive integration of environmental information in the driver’s tactical mental representations of the Cognition Module, according to their physical characteristics (e.g. size, colour, movement) and their saliencies in the road scene for a human eye. The second perceptive process is perceptive exploration (based on Neisser’s theory of perceptive cycle; [129]), which is a “knowledge-driven” process (i.e. top-down integration of perceptive information) in charge to continuously update the driver’s mental models of the Cognition Module, and to actively explore the road scene, according for example to the expectations included in tactical representations.



**Figure 46: Functional architecture of the COSMODRIVE Perception Module**

These perceptive processes of informational search and integration are both under the control of a key mechanism of the Perception Module, the Visual Strategy Manager (VSM). This process is indeed in charge to manage Visual Queries (i.e. information to be obtained) coming from the different cognitive processes that are active at a given time. The visual strategy manager task is to determine the order of priority of these queries and, on this basis, to specify the perceptive exploration strategies for exploring the road scene. Information collected is then transmitted to the querying cognitive processes. Through such a Perception Module, the model is able to dynamically explore the road scene with its virtual eye and to dynamically



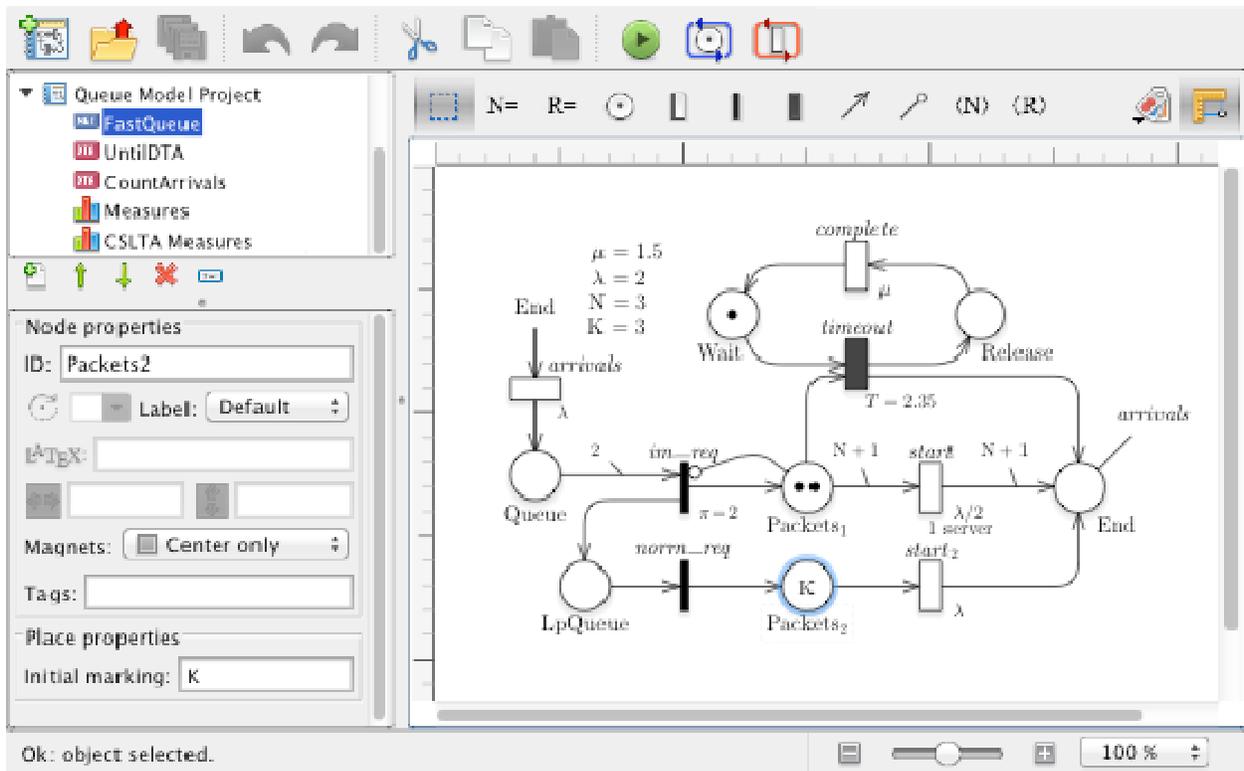
integrate perceptive information. It also possible to simulate drivers' visual distraction (if the model has to observe on-board screen, for instance).

### **3.5.3 GreatSPN for MDPN (UTO)**

#### **3.5.3.1 Introduction**

GreatSPN is a tool developed by the University of Torino in the last 30 years. It is a software framework for the verification of systems, represented with the Petri net formalism, that has been used with success to model many real cases, like bandwidth load in multiprocessor systems, chemical reaction networks, peer-to-peer systems, UML diagrams, and other. The extension to include Markov Decision Process (MDP) solvers and Markov Decision Petri Nets (MDPN) as a Petri net language for the high level definition of MDP is instead work that started a few years back and is it still under development, in particular to adapt it to the needs of HoliDes. Adaptation concerns the graphical user interface (GUI) described in the following, and the MDPN/MDP solvers, described in Section 3.5.3.3

The GUI allows drawing the models graphically, using the Petri net formalism. The interface of the GUI is shown in Figure 47.

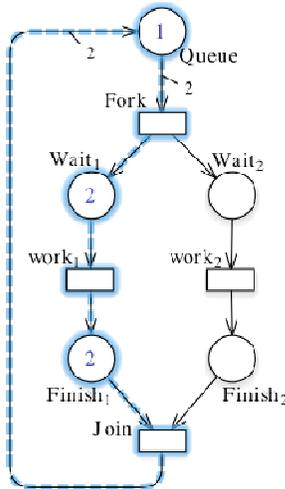


**Figure 47: The GreatSPN graphical interface**

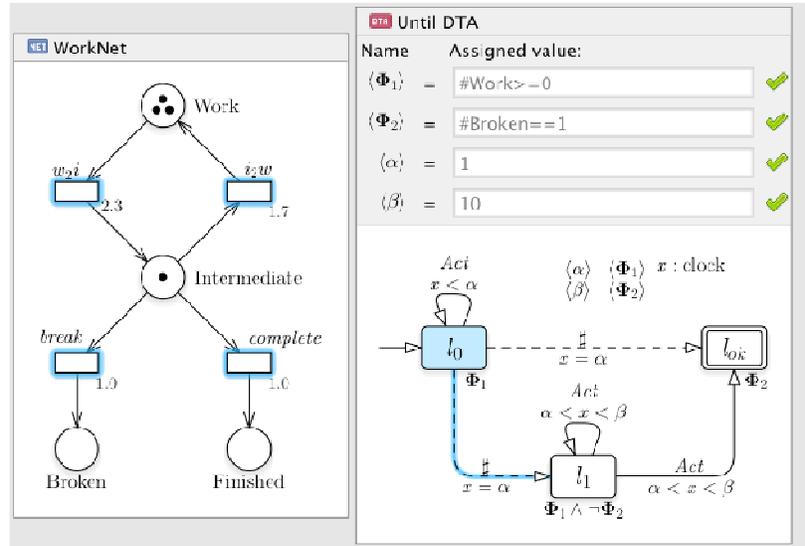
The general data model of the GreatSPN editor is a compositional model where each component is Petri net, or an automaton. Components can then be combined into a larger model using *algebra*, a software element for the composition of Petri Nets which is also part of GreatSPN.

Model design (depicted in the central art of the window) is a fully interactive, WYSISWYG application, where the modeller draws places, transitions, arcs, and the other model elements by a point-and-click approach.

Drawn models can be tested interactively, to better understand the model behaviour, and to identify the invariants. Two examples of interactive testing are shown in Figure 48.



(A) Interactive visualization of a P-semiflow of a GSPN.



(B) Interface of the interactive CSLTA model checking simulation.

**Figure 48: An example of interactive testing in GreatSPN**

Invariant visualization supports P-semiflows and T-semiflows, which characterize the behaviour of the model (A), while interactive simulation (B) allows the user to play with the model, activating its transitions to simulate the behaviour of the system and observe the result.

Once a model has been drawn, performance indices can be computed on it using a collection of numerical solvers. A batch of indices can be specified through the GUI, which invokes the solvers, performs the computation and shows the results interactively. Figure 49 shows the interface for the specification of performance indices on a Petri net model.

Target model: Fork-Join Solver: MC4CSLTA

Template parameters:

Name:	Assigned Value:		
$\langle n \rangle$	<span style="border: 1px solid #ccc; padding: 2px;">ranges</span>	from: <span style="border: 1px solid #ccc; padding: 2px;">1</span> ✓	to: <span style="border: 1px solid #ccc; padding: 2px;">10</span> ✓
		step: <span style="border: 1px solid #ccc; padding: 2px;">1</span> ✓	<input type="button" value="↑"/> <input type="button" value="↓"/>

Solver parameters:

Epsilon: 0.0000001 Linear Solution: GMRES

MRP Method: IMPLICIT Solution method: FORWARD

On the fly solution

Performance indexes are computed in Steady state Transient at: 1.0 ✓

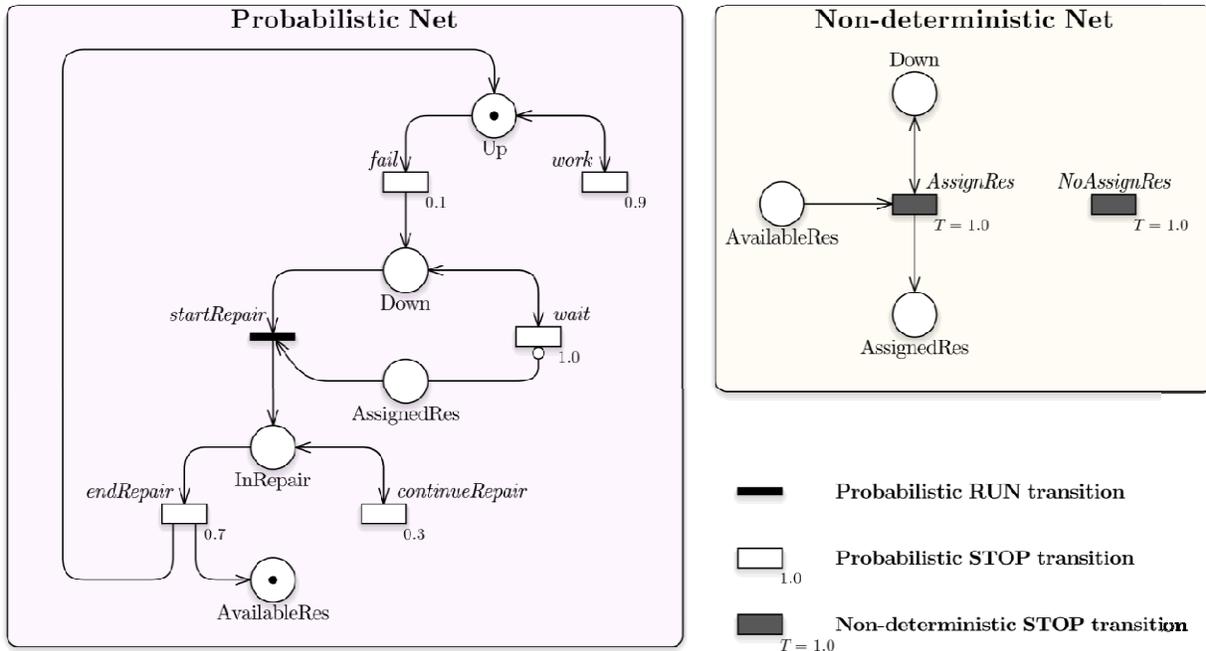
Measures:

Pos:	Measure:		
0° <input type="checkbox"/> <span style="background-color: #333; color: white; padding: 2px;">ALL</span>	All place distributions and transition throughputs.	<input type="button" value="Results..."/>	<input type="button" value="Compute"/>
1° <input type="checkbox"/> <span style="background-color: #ffc000; padding: 2px;">PERF</span>	<span style="border: 1px solid #ccc; padding: 2px;">E{#Queue}</span> ✓ =	<input type="button" value="Results..."/>	<input type="button" value="Compute"/>
2° <input type="checkbox"/> <span style="background-color: #2e7d32; color: white; padding: 2px;">CSLTA</span>	<span style="border: 1px solid #ccc; padding: 2px;">PROB_TA &gt; 0.9 MyDTA (   act0=Fork   Phi0=#Queue&gt;n</span> ✓ =	<input type="button" value="Results..."/>	<input type="button" value="Compute"/>

**Figure 49: Definition of performance indices in GreatSPN**

### Compositionality of MDPN models

The GUI will support compositionality of MDPN models, based on the basic functionality *algebra* of GreatSPN. Two distinct parts compose MDPN models: a *probabilistic net*, and a *non-deterministic net*, both modelled as normal Petri nets in the GUI, as shown in Figure 50, which displays an MDPN model drawn with the GreatSPN GUI.



**Figure 50: An MDPN in GreatSPN**

Compositionality of the two sub-models creates a single model where events can be local to a sub-model, or synchronized between multiple sub-models. The state of the net, represented with the places (circles), can be local (like place *InRepair*) or shared (like place *Down*). The MDPN model can then generate a Markov Decision Process (MDP), which is the underlying statistical process that represents the MDPN behaviour. The full automatic compositionality of MDPN nets is under development, and will be realized under the HoliDes project.

### 3.5.3.2 State-of-the-Art

In the literature, to the best of our knowledge, very few alternative high-level formalisms for MDPs and related tools were proposed.

For instance, models of the probabilistic model checking tool PRISM [98] consist of a number of modules, each of which corresponds to a number of transitions. Each transition is guarded by a condition on the model's variables, and the transitions of a module can update local variables of the module. Multiple transitions may be simultaneously enabled, and the choice between them is nondeterministic; the chosen transition determines a probabilistic choice as to how the variables should be updated. Modules may communicate through synchronization on shared actions with

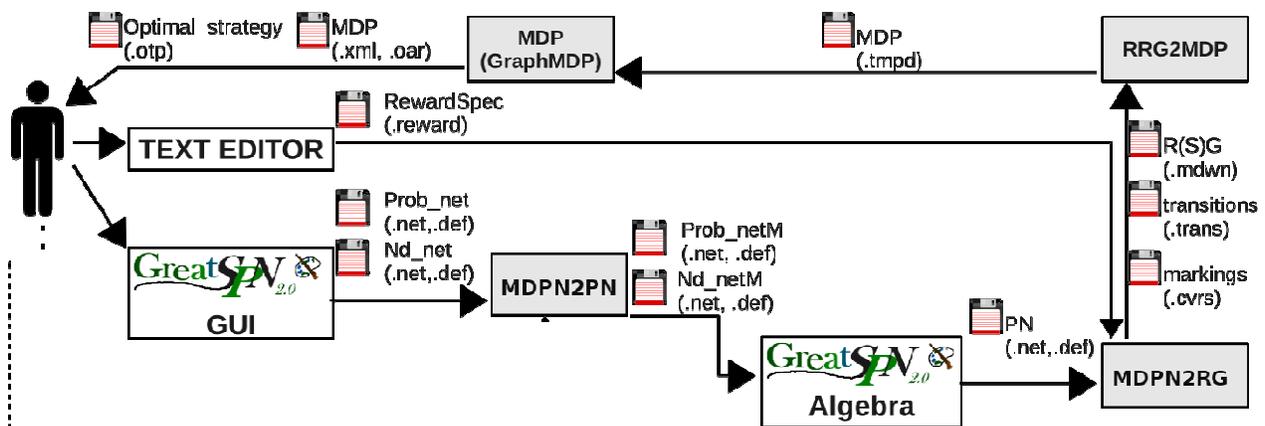
other modules. PRISM does not directly support a multistep nondeterministic or probabilistic transition accounting for the evolution of all components in a given time unit: this can be explicitly modelled by using a variable for each component which records whether the component has taken a transition this time unit.

The modelling language MODEST [22] incorporates aspects from process modelling languages and process algebras, and includes MDPs as one of the many formalisms, which it can express. Stochastic transition systems [5] also subsume MDPs, but also permit both exponentially timed and immediate transitions. Unfortunately they are not supported by a tool.

A number of process algebras featuring nondeterministic and probabilistic choice have been introduced; reader can refer to [84] for an overview of a number of these.

### 3.5.3.3 MTT Description

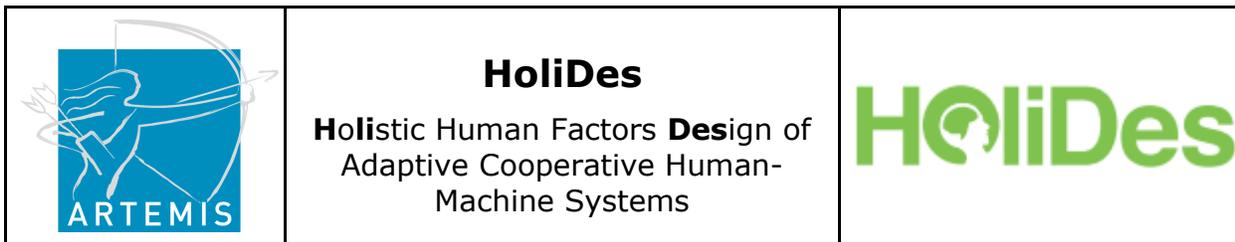
**The MDP module.** The GreatSPN suite of UTO provides a framework to design and solve MDPN models by means of specific modules.



**Figure 51: MDPN solver Architecture**

Indeed these modules transform an MDPN model expressed as a pair of non-deterministic and probabilistic subnets plus a reward function specification into an MDP model and then solve such MDP, deriving an optimal strategy.

The architecture of this MDPN framework is depicted in Figure 51. The user must specify  $PN^{nd}$  and  $PN^{pr}$  subnets (in Figure 50 called **Prob\_net** and **ND\_net**) by means of the *GreatSPN* GUI. A special annotation is used to associate sets of *components* with transitions, and to distinguish between run and stop transitions. Different



priorities can be assigned to transitions: this allows one to avoid useless interleaving when deriving the MDP model, and to force a correct ordering of probabilistic or non-deterministic intermediate (immediate) steps. In addition the **RewardSpec** file must be prepared: it is a textual file where the reward functions to be optimized is specified according to a given grammar.

The transformation process consists of four steps: (1) the non-deterministic and probabilistic subnets are modified by the **MDPN2PN** module that adds some places and two (timed) transitions; (2) the resulting new subnets (**Prob\_netM** and **ND\_netM**) are composed through the *algebra* module of *GreatSPN*; (3) from the obtained PN/WN the (S)RG is generated using the module **MDPNRG**, that produces also two files containing the list of the non-deterministic transition sequences (the MDP actions) and markings description (the MDP states), needed to compute the value of the reward function associated with the MDP states and actions; (4) module **RG2MDP**, generates the final MDP: the states of the MDP correspond to the *tangible* states produced by the previous module, the MDP actions and the subsequent probabilistic transitions, correspond to the *maximal immediate non-deterministic/probabilistic paths* respectively, departing from the non-deterministic/probabilistic tangible markings and reaching probabilistic/non-deterministic tangible markings. In order to make the MDP solution more efficient, the reduction algorithm selects among the actions that connect the same tangible states, that with minimal (or maximal, depending on the optimization problem) reward value. The MDP file is produced in an efficient format which is accepted in input by the **MDP** solver module (based on the *graphMDP* library), that produces the optimal strategy and corresponding optimal reward value.

### 3.5.4 Bayesian Autonomous Driver Mixture-of-Behaviours Models – BAdMoB (OFF)

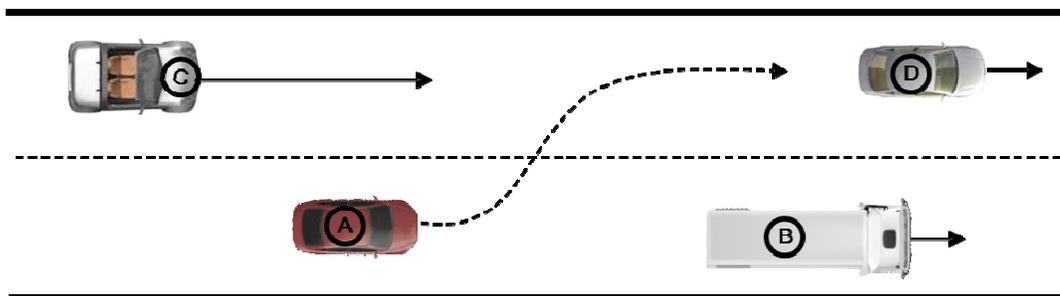
This section shall provide an overview about Bayesian Autonomous Driver Mixture-of-Behaviours (BAD MoB) models. BAD MoB models are human behaviour models based on DBNs (Section 2.4.7) and will be utilized in WP9 to provide an AdCoS application with prediction about the intentions of human drivers. Certain sections have already been reported in D3.3 and D9.3 but will be repeated here in order to provide a more coherent overview.

#### 3.5.4.1 Introduction

As described in D9.3, the AdCoS application for adapted assistance investigated in WP9 is a unique supporting system that shall adapt to the behaviour of the different agents, depending on the internal and external conditions. The AdCoS under

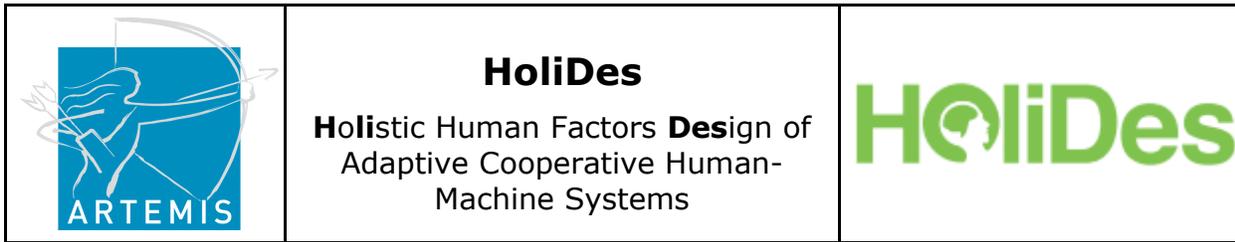
consideration consists of four cars with machine agents and human agents inside (Figure 52) travelling on a highway. In the primary use case for adapted assistance, car A wants to change the lane to overtake truck C. During this manoeuvre, a collision with the other traffic participants has to be avoided. It is assumed that car A will be equipped with several machine agents: a Lane-Change Assistant, an Overtaking-Assistant, and an advanced Forward Collision Warning (FCW) system that provides autonomous assisted and emergency braking functionalities.

Currently, these 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. To give an example, as driver A approaches the lead-vehicle C in order to start the overtaking manoeuvre, he can potentially trigger warnings and possible interventions from the FCW due to the decreasing distance to C. 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 develop a Driver Intention Recognition (DIR) module that provides the different machine agents with predictions about the intentions of the human operator. For this, the DIR module will consult a probabilistic model of the human driver based on previously developed probabilistic driver models, which we call Bayesian Autonomous Driver Mixture-of-Behaviours (BAD MoB) models. As the name suggests, the DIR module solely focusses on the automotive use-cases addressed in WP9. However, the core techniques used are domain-independent and are applicable for other domains and use-cases.



**Figure 52: Representation of the target-scenario (the problem that the AdCoS intends to solve) in AUT domain.**

Intention recognition is primarily concerned with the recognition of behaviour intentions, which are defined as “a person’s intentions to perform various behaviours” [47]. In the automotive domain, i.e., the case of driving intentions, behavioural intentions mainly refer to the intentions of a human driver to follow



certain behavioural schemata or to perform certain driving manoeuvres like e.g., overtaking or lane changes (e.g., [112]).

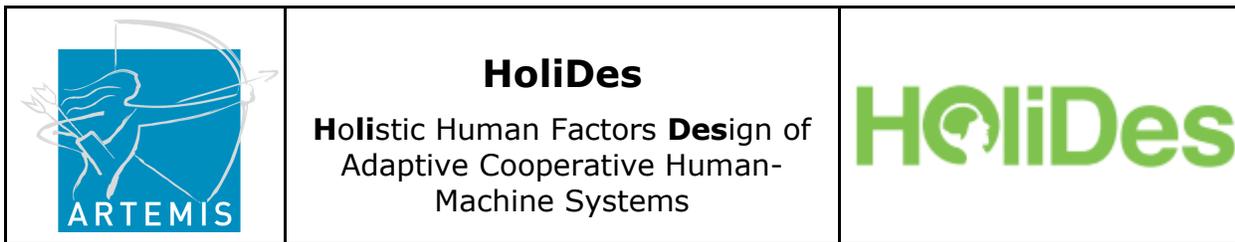
Under the assumption that a person has a sufficient degree of actual control over the intended behaviour (i.e., the corresponding task is not executed by another agent), the existence of an intention implies the readiness for execution and “*people are expected to carry out their intentions when the opportunity arises*” [4]. Following these implications, an intention can be seen as an immediate antecedent or predictor of the human’s behaviour in the nearest future [4]. Knowledge about the current driving intentions of the human driver would therefore allow an AdCoS to adapt in order to comply these intentions and therefore decrease the risk of decreased user-acceptance or to initiate appropriate countermeasures when these intentions do not comply with the assessed situation.

Intentions are theoretical constructs that cannot be measured or assessed directly [91]. This is especially true in the case of driving, where the choice and execution of manoeuvres may be highly automated skills whose execution will not necessarily be considered by the driver as intentional [4]. Accordingly, they have to be inferred from the available context.

### **3.5.4.2 State-of-the-Art**

The modelling of human driving behaviour has been an extensive area of research in the domain of transportation systems. A driver can be seen as a human agent whose skills can be described by three stages labelled the cognitive, associative, and autonomous layers [6]. At the cognitive layer, the general planning of a journey is handled. For example, the driver chooses the route and transportation mode, and evaluates resulting costs and time consumption. At the associative layer, the driver has to plan manoeuvres, allowing him/her to negotiate the “right now” prevailing circumstances, for instance turning at an intersection or accepting a gap. Finally, at the autonomous layer, the driver has to execute sequences of actions that together form a manoeuvre. Examples are braking manoeuvres in order to keep a safe distance or turning the steering wheel to remain in the middle of the lane. According to these stages, various modelling approaches seem to be adequate: production-system (e.g. models in the ACT-R-, SOAR-, and CASCaS-architectures) for the cognitive and associative stages [150][151][116][1] and control-theoretic models for the autonomous stage [86][171][25][3]. These kinds of models are quite standard approaches now [27]. More recently, approaches for the autonomous and associative stage have been broadened by probabilistic driver models (e.g., [48][96] [126][43][89]).

Driver models have been used in traffic scenario simulations to provide safety assertions and support risk-based design [27][25]. However, with the need for



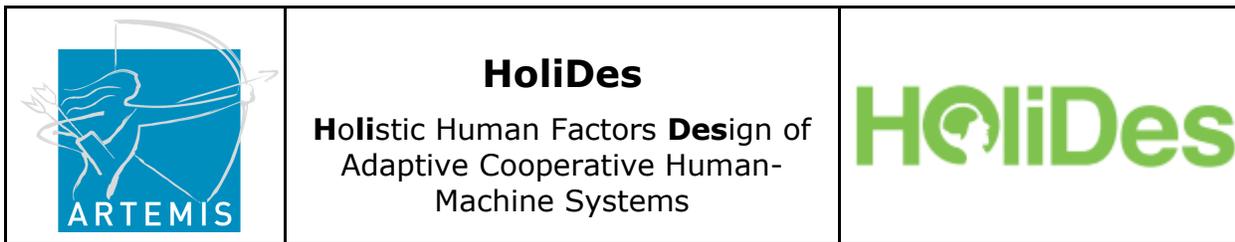
smarter and more intelligent assistance, the problem of transferring human skills into the envisioned technical systems is becoming more and more apparent [183], and the focus is shifting towards the utilization of driver models within assistance systems. Especially the use of driver models for the recognition and prediction of driver's behaviours and intention has gained great attention in current research. The ability to predict the driver's manoeuvre intentions is considered a key elemental technology for the future generation of assistance systems for both safety and eco-driving [10][88][118][127][135], and in the last years, several studies on recognition of driver's intentions have been reported [118], addressing the recognition of lane-change intentions [20][38][124][149], braking actions [123][122], turning manoeuvres [111][38], and overall trajectory prediction [174].

Driver models that shall be used in real-world applications like assistance systems must be able to deal with uncertain or noisy information. Therefore, they are most commonly based on approaches for neuro-/fuzzy- and especially probabilistic models like, e.g., Hidden Markov Models (HMMs) and (Dynamic) Bayesian Networks (DBNs) [126][20][124][2][88][87][13][135][152][10]. Current models for intention prediction are sufficient to predict single specific behaviours (e.g., lane-changes or braking manoeuvres) of the human driver up to approx. three seconds [127][135][38]). An additional overview of the state of the art for intention recognition can be found e.g., in [91] and [23].

Due to the variability of human cognition and behaviour, the irreducible lack of knowledge about underlying cognitive mechanisms, and the irreducible incompleteness of knowledge about the environment, we will focus on the use DBNs, based on previously developed BAD MoB models. Prior to HoliDes, BAD MoB models were solely developed and used as probabilistic driver models for autonomous control in simulator environments. We have developed machine-learning methods to learn the parameters and graph-structure of BAD MoB models in respect to the pertinent perceptual feature needed to mimic human driving behaviour using a set of psychological motivated percepts that have been proposed in the literature. For HoliDes, we are working on extending BAD MoB models to make them a valuable tool for intention recognition.

### **3.5.4.3 MTT Description**

In this section, we will give an overview about the general structure of BAD MoB models for intention recognition. We'd like to emphasize that most of the work presented relies on theoretical assumptions based on the nature of the Automotive domain that have yet to be validated in respect to actual experimental data, which is expected to be gathered early in 2015. Until then, we focus on the development



of general template structures, whose exact structure can be refined in the light of experimental data.

### 3.5.4.3.1 Variable Selection

Within the context of HoliDes, a BAD MoB model for intention recognition defines a (conditioned) JPD over sets of discrete and continuous random variables representing *intentions*, *behaviours*, (*human*) *actions*, and (*context*) *observations*. In order to select a set of intentions of interest, it is required to select a set of behaviours/manoeuvres we expect the driver to perform. Based on the use-cases for adapted assistance (see D9.3), we selected the following set of behaviours that would allow the human driver to travel on a highway in the absence of emergencies:

- To perform a lane change to the left lane
- To perform a lane change to the right lane
- To perform lane-following
- To perform car-following

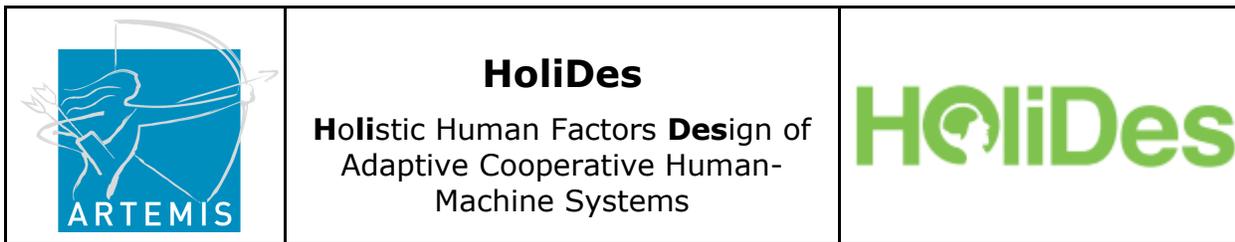
Within a BAD MoB model, these different behaviours/manoeuvres are represented by a discrete random variable  $B$ , with the possible values

$Val(B) = \{\text{lane change left, lane change right, lane-following, car-following}\}$ .

Not all of these behavioural schemes or manoeuvres are necessarily triggered intentionally, e.g., under the assumption of normative driving, a transition from lane-following to car-following should occur naturally given the current situation, if no countermeasure (like e.g., performing a lane-change in order to overtake) is initiated by the driver. Furthermore, given the highly dynamic environment in the automotive use-cases and the limited knowledge about the environment due to limited sensor capabilities (e.g., surrounding traffic participants may be outside of the detection range, or a leading vehicle may occlude a second leading vehicle), we limit intention recognition to “short-term” intentions, which in the context of the selected use-cases can be narrowed down to lane change intentions. By now, we therefore selected two distinct behaviour intentions and an additional absence of intention:

- The intention to change to the left lane
- The intention to change to the right lane
- The absence of the above intentions

The selected behavioural intentions of the driver are represented by a discrete random variable  $I$ , with the possible values



$Val(I) = \{\text{lane change left, lane change right, default}\}$ .

Additional or more fine-grained behaviours (following the task analysis described in D9.3) and intentions may be added after an evaluation of experimental data, which is expected early 2015.

Based on exemplary datasets provided by CRF, the use-cases for adapted assistance and the system specification for the AdCoS for adapted assistance (see D9.3), we then consider the following actions, represented by a set of discrete and continuous random variables  $A = \{A_1, \dots, A_n\}$ :

Variable	Type	Description
<i>Steering angle</i>	Continuous	Steering angle value
<i>Acceleration</i>	Continuous	Acceleration of the driver's vehicle
<i>Head position</i>	Continuous	Position of the driver's head
<i>Head position rate of change</i>	Continuous	Rate of change of the driver's head position
<i>Head orientation</i>	Continuous	Orientation of the driver's head (yaw, roll, pitch)
<i>Head orientation rate of change</i>	Continuous	Rate of change of the driver's head orientation
<i>Direction Indicator signal</i>	Discrete	Status of the left and right indicators

All available internal and external context information that does not correspond to the actions of a driver is represented by a set of discrete and continuous random variables denoted by  $\phi = \{\phi_1, \dots, \phi_m\}$ . It is expected that most input available for intention recognition is already pre-processed, i.e., information about the environment is not provided as raw sensor data but instead as filtered (point) estimates based on an internal world model inherited by the sensors itself (e.g., by the use of Kalman-Filters). As a consequence, a BAD MoB model does not utilize a hidden world model that needs to be estimated from noisy sensor data, and can instead utilize the provided estimates as evidence. While many observation variables correspond to available sensor data (c.f., D9.3), additional variables are defined as functions of this values, e.g., rates of changes, time headway, or time-to-contact values. By now, we focus on the following variables:

Input	Type	Description
<i>Lane curvature</i>	Continuous	Curvature of the road
<i>Lateral derivation</i>	Continuous	Lateral distance between the middle of the lane and the longitudinal axis of the driver's vehicle
<i>Yaw angle</i>	Continuous	Angle between the longitudinal axis of the driver's vehicle and lane direction, tangent to the lane (also called "lane yaw angle")
<i>Yaw angle rate of change</i>	Continuous	Rate of change of the yaw angle
<i>Lead car lat. speed</i>	Continuous	Lateral velocity of the lead vehicle

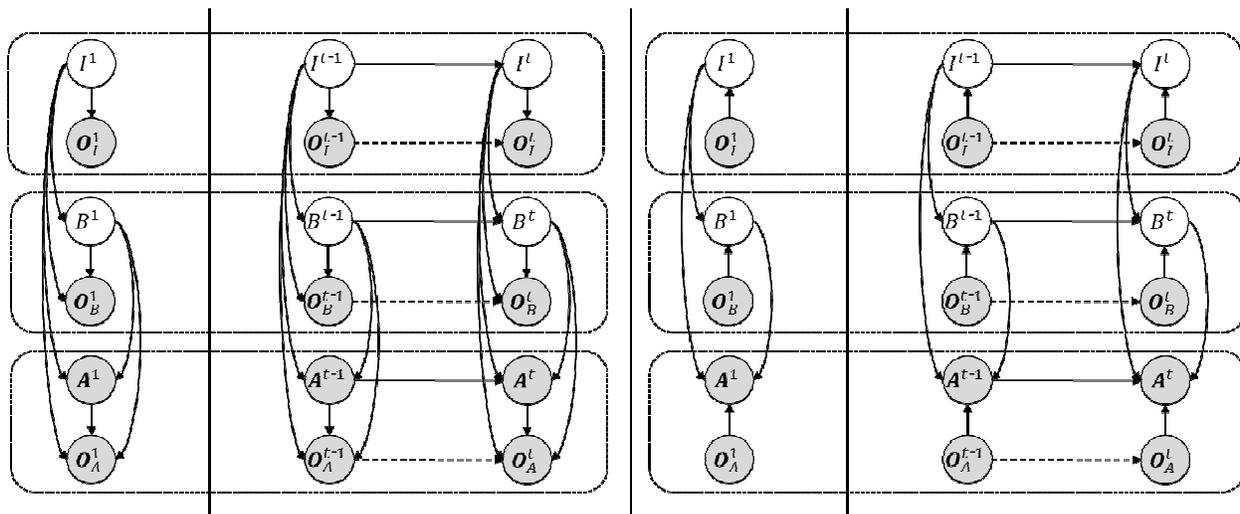
<i>Lead car lat. acceleration</i>	Continuous	Lateral acceleration of the lead vehicle
<i>Lead car long. speed</i>	Continuous	Longitudinal velocity of the lead vehicle
<i>Lead car long. acceleration</i>	Continuous	Longitudinal acceleration of the lead vehicle
<i>Lead car lat. distance</i>	Continuous	Lateral distance of the lead vehicle in respect to the driver's vehicle
<i>Lead car long. distance</i>	Continuous	Longitudinal distance of the lead vehicle in respect to the driver's vehicle
<i>THW to lead car</i>	Continuous	Time headway to the lead vehicle
<i>THW to lead car change of rate</i>	Continuous	Rate of change of the time headway to the lead vehicle
<i>TTC to lead car</i>	Continuous	Time-to-contact to the lead vehicle
<i>TTC to lead car rate of change</i>	Continuous	Rate of change of the time-to-contact to the lead vehicle
<i>Velocity difference</i>	Continuous	Difference between the velocities of the driver's and the lead vehicle
<i>Velocity difference rate</i>	Continuous	Rate of change of the difference between the velocities of the driver's and the lead vehicle
<i>Velocity</i>	Continuous	Velocity of the driver's vehicle
<i>VDD</i>	Discrete	Visual Distraction Detection
<i>VTSD</i>	Continuous	Visual Time Sharing Distraction

In general, it is assumed that both actions and observations are always *observable* (i.e., the DIR module will be provided with actual values that can be used as evidence during inference), while intentions and behaviours are always *hidden*. The distinction between actions and observations is therefore rather arbitrary, as both will be provided by dedicated sensors. However, as a first step, we will restrict the inclusion of temporal dependencies to dependencies between actions. These restrictions will be relaxed during the course of the project, which consequently may render variables representing rate-of-changes redundant. Additionally, not all available selected variables are necessarily valuable for intention recognition and accordingly not all of them will necessarily be included in the actual BAD MoB models used for intention recognition.

### 3.5.4.3.2 Model Structures

Concerning the state of the art, both generative and discriminative approaches have been proposed and successfully used for intention recognition. In respect to the variables defined above, the generative approach can be seen as the task to find a factorization (i.e., the graph-structure) and corresponding parameters for the JPD  $P(I^{1:T}, B^{1:T}, A^{1:T}, O^{1:T})$ , while the discriminative approach results in the task to find a factorization and corresponding parameters for the conditional JPD  $P(I^{1:T}, B^{1:T}, A^{1:T} | o^{1:T})$ . In HoliDes, we investigate both alternatives, for which we will

give first theoretical results in the form of two general “template” structures, shown in Figure 53. Both templates define a high-level factorization of their JPDs that ensures efficient inference using standard exact inference techniques, while allowing many different finer factorizations of its CPDs. The final factorization of the BAD MoB model and the parameters of the (conditional) probability distributions will be derived by machine-learning methods from multivariate time series of human behaviour traces, once they are available (expected early 2015).



**Figure 53: Template structures of BAD MoB models**

These template structures of BAD MoB models for intention recognition, (loosely) based on FHMMs (left) and HMDTs (right), both defined by a Bayesian network for the first time-slice  $t=1$  and a 2TBN for all  $t>1$ . Blank nodes represent hidden variables; shaded nodes represent variables that are assumed to always be observed. Dotted lines imply optional temporal dependencies between observations. Dotted boxes imply the scope of component-models with private observations.

### 3.5.4.3.2.1 Generative Approach

For the generative approach we need to define a factorization of the JPD  $P(I^{1:T}, B^{1:T}, A^{1:T}, O^{1:T})$ . The chain rule of probabilities [92] allows without any independency assumptions to factorize the JPD as:

$$\begin{aligned}
 &P(I^{1:T}, B^{1:T}, A^{1:T}, O^{1:T}) \\
 &= P(I^1, B^1, A^1, O^1) \prod_{t=2}^T P(I^t, B^t, A^t, O^t | I^{1:t-1}, B^{1:t-1}, A^{1:t-1}, O^{1:t-1})
 \end{aligned}$$



## HoliDes

Holistic Human Factors Design of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

In order to make inferences computational tractable, we rely on the common assumptions for temporal models that the system under consideration can be approximated as stationary dynamic Markovian. The Markov assumption states that the future is conditionally independent of the past, given the present, and allows us to define a more compact representation of the JPD:

$$P(I^{1:T}, B^{1:T}, A^{1:T}, O^{1:T}) \\ = P(I^1, B^1, A^1, O^1) \prod_{t=2}^T P(I^t, B^t, A^t, O^t | I^{t-1}, B^{t-1}, A^{t-1}, O^{t-1}).$$

The assumption of stationary processes then allows us to use a single 2TBN  $P(I^t, B^t, A^t, O^t | I^{t-1}, B^{t-1}, A^{t-1}, O^{t-1})$  for all  $t > 0$ . We emphasize that both assumption most certainly do not hold for the complex human driving behaviour. However, said assumption pose a necessary restriction in order to make inference computational tractable that most certainly cannot be relaxed.

Given these, the chain rule of probabilities allows us, without any further assumptions, to factorize the CPD  $P(I^t, B^t, A^t, O^t | I^{t-1}, B^{t-1}, A^{t-1}, O^{t-1})$  as:

$$P(I^t, B^t, A^t, O^t | I^{t-1}, B^{t-1}, A^{t-1}, O^{t-1}) \\ = P(I^t | I^{t-1}, B^{t-1}, A^{t-1}, O^{t-1}) P(B^t | I^{t-1:t}, B^{t-1}, A^{t-1}, O^{t-1}) \\ P(A^t | I^{t-1:t}, B^{t-1:t}, A^{t-1}, O^{t-1}) P(O^t | I^{t-1:t}, B^{t-1:t}, A^{t-1:t}, O^{t-1}).$$

Starting from this factorization, we make the following additional independency assumptions (which will be tested once experimental data is available):

- $P(I^t | I^{t-1}, B^{t-1}, A^{t-1}, O^{t-1}) = P(I^t | I^{t-1})$ : Given only the previous intention  $I^{t-1}$ , the current intention  $I^t$  of the human operator is conditionally independent of the previous behaviour  $B^{t-1}$ , actions  $A^{t-1}$ , and observations  $O^{t-1}$ . As intentions can be seen as the antecedent of behaviour, and manoeuvres imply the use of specific sensor-motor patterns, these two first assumptions seem reasonable. In contrast, the third independency assumption is stated for computational reasons. However, we will investigate the influence of observations for directly predicting the evolution of intentions in the discriminative approach.
- $P(B^t | I^{t-1:t}, B^{t-1}, A^{t-1}, O^{t-1}) = P(B^t | I^t, B^{t-1})$ : Given the current intention  $I^t$  and the previous behaviour/manoeuvre  $B^{t-1}$ , the current behaviour  $B^t$  is conditionally independent of the previous intention  $I^{t-1}$ , actions  $A^{t-1}$ , and observations  $O^{t-1}$ . Once again, under the assumption that intentions are the antecedent of behaviours, given  $I^t$ , we should not gain any additional knowledge from  $I^{t-1}$  concerning  $B^t$ . Furthermore, given  $B^{t-1}$ , we should not



## HoliDes

Holistic Human Factors Design of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

gain additional knowledge from  $A^{t-1}$ . Once again, the main controversy lies in the conditional independence from  $O^{t-1}$ .

- $P(A^t | I^{t-1t}, B^{t-1t}, A^{t-1}, O^{t-1}) = P(A^t | I^t, B^t, A^{t-1})$ : Given the current intention  $I^t$ , the current manoeuvre  $B^t$  and the previous actions  $A^{t-1}$ , the current actions  $A^t$  are conditionally independent of the previous intention  $I^{t-1}$ , the previous behaviour  $B^{t-1}$ , and the previous observations  $O^{t-1}$ . Under the assumption that the drivers actions are triggered by his intentions (e.g., observing the side-view mirror when intending to perform a lane-change) and the current manoeuvre we should not gain additional knowledge from the former intentions or behaviours. Concerning the independency assumption for the previous observations, we will investigate the influence of observations for directly predicting the evolution of actions in the discriminative approach.
- $P(O^t | I^{t-1t}, B^{t-1t}, A^{t-1t}, O^{t-1}) = P(O^t | I^t, B^t, A^t, (O^{t-1}))$ : Given the current intention  $I^t$ , behaviour  $B^t$ , and actions  $A^t$  (and potentially the previous observations  $O^{t-1}$ ), the current observations  $O^t$  are conditionally independent from the previous intention  $I^{t-1}$ , behaviour  $B^{t-1}$ , and actions  $A^{t-1}$ . By now, we will not include temporal dependencies between observations, however, this assumption will be thoroughly tested in the light of experimental data and additional dependencies will be added if necessary. Additionally, based on previous experience, we expect that we can find additional independencies that allow us to factorize the CPD  $P(O^t | I^t, B^t, A^t, (O^{t-1}))$  as  $P(O_I^t | I^t, (O_I^{t-1}))P(O_B^t | I^t, B^t, (O_B^{t-1}))P(O_A^t | I^t, B^t, A^t, (O_A^{t-1}))$ , i.e., independent sets of observations relevant for intentions, behaviours, and actions.

To summarize these assumptions, we will assume that the JPD  $P(I^{1:T}, B^{1:T}, A^{1:T}, O^{1:T})$  can be factorized as:

$$\begin{aligned}
 &P(I^{1:T}, B^{1:T}, A^{1:T}, O^{1:T}) \\
 &= P(I^1)P(O_I^1 | I^1)P(B^1 | I^1)P(O_B^1 | B^1, I^1)P(A^1 | B^1, I^1)P(O_A^1 | A^1, B^1, I^1) \\
 &\prod_{t=2}^T P(I^t | I^{t-1})P(O_I^t | I^t)P(B^t | B^{t-1}, I^t)P(O_B^t | B^t, I^t)P(A^t | A^{t-1}, B^t, I^t)P(O_A^t | A^t, B^t, I^t).
 \end{aligned}$$

The graph structure of this template is shown in Figure 53 (left), which can be seen as a modification of a model class known as Factorial Hidden Markov Models (FHMM) [54].

Concerning the finer factorization of  $P(A^t | A^{t-1}, B^t, I^t)$ , we will start our modelling efforts by assuming independent Markov chains for each action, i.e., we assume that we can ignore the hopefully loose coupling between the different actions, which results in  $P(A^t | A^{t-1}, B^t, I^t) = \prod_{i=1}^n P(A_i^t | A_i^{t-1}, B^t, I^t)$ . Unfortunately, in general, we can't make the same assumptions about the environment, i.e.,

$P(\mathbf{O}^t | (\mathbf{O}^{t-1}), \mathbf{A}^t, \mathbf{B}^t, I^t) \neq \prod_{j=1}^m P(o_j^t | (o_j^{t-1}), \mathbf{A}^t, \mathbf{B}^t, I^t)$ . The resulting need to provide a reasonable factorization for the observations makes the use of generative models rather complicated. On the other hand, there exist a large amount of well-documented and efficient techniques to estimate the parameters and structure of these models. We will therefore start with quite strong assumptions concerning the factorization of  $P(\mathbf{O}^t | (\mathbf{O}^{t-1}), \mathbf{A}^t, \mathbf{B}^t, I^t)$  that will primarily be guided by the need for computational efficiency and robust parameter estimation.

### 3.5.4.3.2.2 Discriminative Approach

As we can assume that the observations are always known, we will additionally investigate the use of a discriminative approach, where the model defines the conditional JPD  $P(I^{1:T}, \mathbf{B}^{1:T}, \mathbf{A}^{1:T} | \mathbf{o}^{1:T})$ . Applying the same independency assumptions as discussed for the generative approach, we obtain the following factorization:

$$\begin{aligned}
 P(I^{1:T}, \mathbf{B}^{1:T}, \mathbf{A}^{1:T} | \mathbf{o}^{1:T}) &= P(I^1 | \mathbf{o}_1^1) P(\mathbf{B}^1 | I^1, \mathbf{o}_B^1) P(\mathbf{A}^1 | \mathbf{B}^1, I^1, \mathbf{o}_A^1) \\
 &\prod_{t=2}^T P(I^t | I^{t-1}, \mathbf{o}_I^t) P(\mathbf{B}^t | \mathbf{B}^{t-1}, I^t, \mathbf{o}_B^t) P(\mathbf{A}^t | \mathbf{A}^{t-1}, \mathbf{B}^t, I^t, \mathbf{o}_A^t).
 \end{aligned}$$

The graph structure of this template is shown in Figure 53 (right), which can be seen as a modification of a model class known as Hidden Markov Decision Trees (HMMDT) [85].

Apparently, the biggest advantage of the above factorization lies in the fact that we don't need to distinctively model the observation sequence  $\mathbf{o}^{1:T}$ , and therefore don't have to make any independency assumptions concerning the nature of the observations. Furthermore, given a rich set of observation variables (including rate of changes), we can reasonable assume conditional independence of intention, behaviours, and actions from the previous observations  $\mathbf{o}^{t-1}$  given the current observations  $\mathbf{o}^t$ . Note that although we assume that actions are always observable, we will explicitly include them in our discriminative model  $P(I^{1:T}, \mathbf{B}^{1:T}, \mathbf{A}^{1:T} | \mathbf{o}^{1:T})$ . This is due to the fact that we plan to answer probability queries over actions, e.g., by computing the likelihood of a sequence of actions  $P(\mathbf{A}^{1:t} | \mathbf{a}^{1:t-1}, \mathbf{o}^{1:t})$ . By implication this also means that we will not aim to answer probability queries about observations (e.g., predicting future observations).

Concerning a finer factorization of  $P(\mathbf{A}^t | \mathbf{A}^{t-1}, \mathbf{B}^t, I^t, \mathbf{o}_A^t)$ , we will once again start with the assumption of independent Markov chains, i.e.,

$P(A^t | A^{t-1}, B^t, I^t, o_A^t) = \prod_{i=1}^n P(A_i^t | A_i^{t-1}, B^t, I^t, o_A^t)$ , where we will try to model each CPD  $P(A_i^t | A_i^{t-1}, B^t, I^t, o_A^t)$  by a conditional sub-network  $\frac{1}{Z} P(A_i^t, o_A^t | A_i^{t-1}, B^t, I^t)$ , where  $Z$  denotes a normalization factor  $Z = \sum_{a_i \in A_i} P(a_i, o_A^t | A_i^{t-1}, B^t, I^t)$ , in the case of a discrete variable  $A_i$ , or resp.  $Z = \int_{-\infty}^{\infty} P(a_j, o_A^t | A_j^{t-1}, B^t, I^t) da_j^t$  in the case of a continuous variable  $A_j$ .

A severe disadvantage of the discriminative approach lies in the fact that the parameters cannot be estimated independently, and consequently not efficiently enough for structure-learning. We will try to soften this disadvantage by the use of approximate techniques proposed by [158].

#### 3.5.4.3.2.3 Utilization

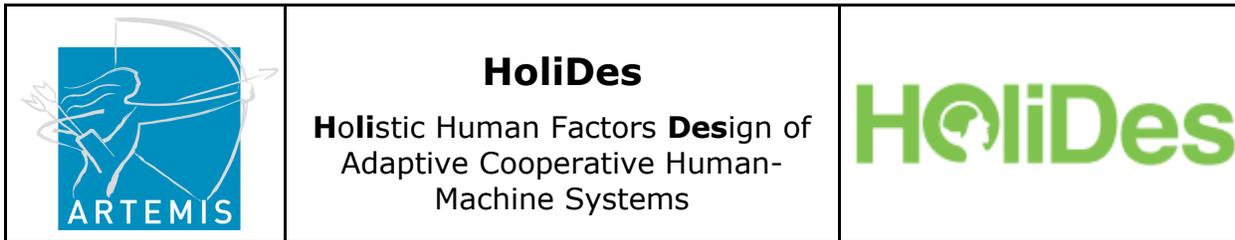
Given a fully specified BAD MoB model, it can be used to constantly (i.e., at each time step  $t$ ) infer the joint belief state of intentions and behaviours given all available evidence about actions and observations obtained so far:  $P(I^t, B^t | a^{0:t}, o^{0:t})$ . Given this joint belief state, we can easily derive the marginal belief states of intentions  $P(I^t | a^{0:t}, o^{0:t})$  and behaviours  $P(B^t | a^{0:t}, o^{0:t})$ . The estimation of the belief state is known as filtering and can, for both template structures, be solved in *constant* time by recursively computing  $P(I^t, B^t | a^{0:t}, o^{0:t})$  from the previous belief state  $P(I^{t-1}, B^{t-1} | a^{0:t-1}, o^{0:t-1})$ .

### 3.5.5 Driver Distraction Classifier (TWT)

#### 3.5.5.1 Introduction

Distraction during driving leads to a delay in recognition of information that is necessary to safely perform the driving task [144]. Thus, distraction is one of the most frequent causes for car accidents [12], [67]. Four different forms of distraction are distinguished while they are not mutually exclusive: visual, auditory, bio-mechanical (physical), and cognitive distraction. Human attention is selective and not all sensory information is processed (consciously). When people perform two complex tasks simultaneously, such as driving and having a demanding conversation, the brain shifts its focus. This kind of attention shifting might also occur unconsciously. Driving performance can thus be impaired when filtered information is not encoded into working memory and so critical warnings and safety hazards can be missed [164]. Sources for distraction of the driver can be located within and outside of the car.

A computational and empirical cognitive distraction model will be developed in order to analyse different signals from in-car measures with the purpose to detect the



distraction degree of the driver. For assessing predictive parameters for cognitive distraction during driving, we run several experiments using a driving simulation and comparing parameters between concentrated driving and distracted driving induced by secondary tasks like conversations or calculation tasks. These measures will include an acoustic analysis including, e.g. the detection of the number of speakers, the degree of emotional content, and information about the driver's involvement in the conversation (e.g., whether the driver himself is speaking). In addition, face-tracking signals such as the blinking of the eyes, head pose and mouth movements will add to the reliability of distraction prediction.

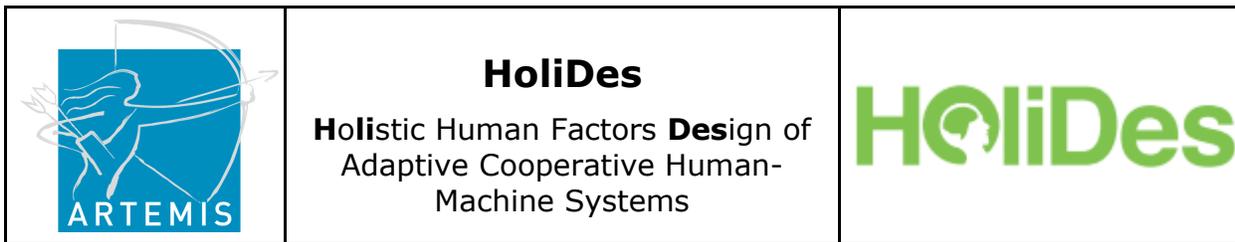
On the one hand, we hope to get new insights about the correlation between auditory signals inside the car and cognitive distraction of the driver from our experimental results. On the other hand, the overall aim for the application of the cognitive distraction model is the development of a mobile user profile computing the individual distraction degree and being applicable also to other systems.

### **3.5.5.2 State-of-the-Art**

Identifying cognitive distraction is more complex than visual distraction because the mechanisms involved in cognitive distraction have not been as precisely described. The detection of cognitive distraction could presumably be assessed best through an integration of a number of different parameters like eye and face measures of the driver (e.g., blink frequency, fixation duration, mouth movements), driving performance measures (e.g., steering wheel movements and breaking behaviour), and as we propose here also auditory signals. Several models for cue integration have been suggested for cognitive modelling of distraction. Support Vector Machines (SVMs) and Bayesian Networks have successfully identified the presence of cognitive distraction using eye movements and driving performance [106], [109]. The recent dynamic Bayesian model by Liang and Lee [107] consists of a combined supervised and unsupervised learning approach. In HoliDes, we will extend this model with higher-level conversational cues, like the degree of estimated conversational interaction as a likely distraction measure.

### **3.5.5.3 MTT Description**

The distraction estimation tool bears the potential to be used online to classify the driver's distraction not only during testing of a prototype, but also during everyday interaction with the AdCoS. This online measure of distraction could in turn be used to adapt the degree of automation of the AdCoS to the driver's state. The cognitive distraction model can be integrated into the following WP9 AdCoS systems: the TAK Simulator AdCoS, the IAS Test Vehicle, and potentially the CRF Test Vehicle. Within



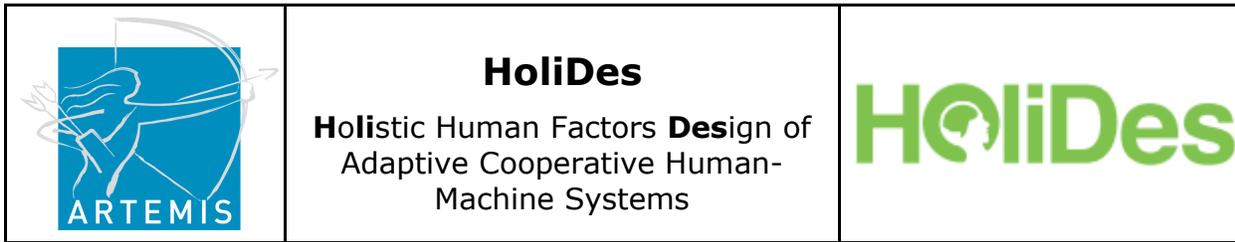
these AdCoS' the distraction signal is used for triggering the adaptation of the system with respect to the driver's attentional status. When the driver is distracted, the system will warn the driver giving a warning tone (TAK AdCoS). Is the car autonomously driving and the driver is distracted, the system will not perform manoeuvres that enhance the risk that the driver needs to intervene. Thus, the system will e.g. not induce overtaking manoeuvres and will instead keep more distance to the pace car (IAS AdCoS). A detailed description of those systems can be found in *D9.2*.

In addition, for integration of the tool into the HF-RTP, its usage during system validation phase plays an essential role. When validating a system it can be very valuable to derive knowledge about the human. While interacting with a prototype or some modules of the AdCoS, the operator's degree of distraction can be determined. Thus, in this context the tool provides feedback whether or not a new system (module) increases or decreases the operator's degree of distraction.

In-vehicle information is needed as input. This includes, but it is not limited to in-car audio recordings, face-tracking data from the driver, and behavioural driving parameters. Audio data involve, e.g., the number of speakers, the amplitude of the noise, speech durations and pauses. By combining face-tracking signals like mouth movements of the driver, it can be identified whether the driver himself is speaking or a co-passenger. Other face-tracking data like the blinking rate might give hints about the distraction degree of the driver, since it was found that cognitive distraction increases the blinking rate [147]. Behavioural driving parameters as another source for inducing distraction of the driver can be used, as e.g. the distance to the pace car increases when the driver is distracted. These data will be weighted according to their technical quality and to their correlating strength for distraction, and will further be integrated in order to compute a temporal distraction degree of the driver. Thus multimodal data integration and synchronization needs to be guaranteed.

The tool provides an online continuous measure of the distraction degree provided by a regression analysis, meaning that the distraction degree will be identified using a time window of about 3-10 seconds of the past. The metrics used to quantify the driver's distraction based on in-car information are developed in T5.2. The different measurements will be integrated in RTMaps provided by INT.

Figure 17 in section 2.4.5 shows the architectural design of the simplified cognitive distraction estimation model. We take the driver's auditory and visual perception into consideration and compute his/her distraction degree based on a resource allocation model. This model from Wickens (2002) [155] states that the more a secondary task takes up the same or similar sensory modalities (auditory vs. visual), codes (visual vs. spatial) and processing stages (perceptual, cognitive,



responses), the more the secondary task leads to distraction from the primary task. The measured parameters derived from in-car audio recordings, face-tracking information of the driver, behavioural car information (e.g. driving parameters) and environmental information like the distance to the pace car to be followed will lead to conclusions about the allocation of the driver's resources and therefore enable the computation of his distraction degree.

### **3.5.6 Pilot Pattern Classifier (TEC)**

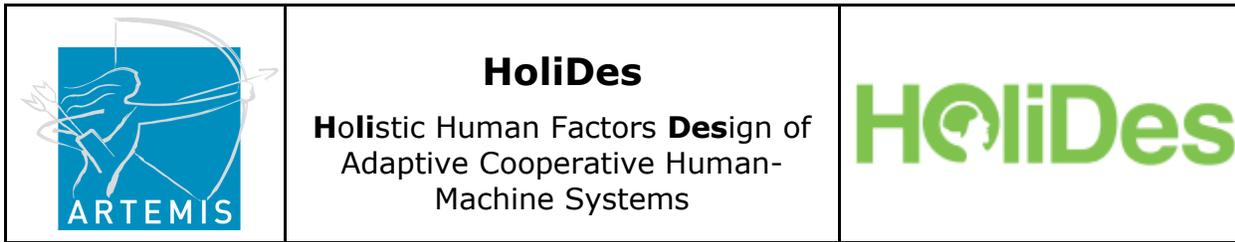
#### **3.5.6.1 Introduction**

Aircraft pilots have to operate more complex vehicles, and therefore go through a strict training programme before getting their flying license. Modern glass cockpits look tidy from the outside and are designed to be as intuitive as possible, but a complex system is functioning behind the scenes. As a result, it becomes increasingly challenging for pilots to fully and continuously manage the display systems of the new models of modern aircrafts, such in the evolution occurred from the MD-80 to the Airbus 330 cockpit. A proper understanding of the relevant information among the many presented on the cockpit is crucial for the pilot in emergency situations in which the time available for understanding the problem could be very short. Fortunately, both situations are rarely encountered in actual flights due to the excellent reliability of aircraft and on board systems. However, when they do occur, the pilot's mental state – a construct including situation awareness, mental workload and fatigue – plays a crucial role in solving the problems.

As is evident from the quoted accident statistics and illustrated by this case [40], the flight crew is still the most commonly contributing factor in fatal accidents worldwide. Also, an ineffective pilot mental state (e.g., peak workload, lack of situation awareness, fatigue, and drowsiness) plays a role in the sequence of events leading to many of these accidents. Therefore, the need for a continuously improved understanding of pilot behaviour and how to optimise crew performance is particularly important.

A pilot's behaviour can be measured; it is possible to collect neurophysiological measures/signals (electroencephalogram-EEG, electrooculogram-EOG, variation of the heart rate by the electrocardiogram-EKG, etc.) that assess the activity of the central and the autonomous nervous system of the pilot during the analysed driving tasks.

Until now several machine learning techniques, such as artificial neural networks (ANN), hierarchical Bayes model, support vector machines (SVM), etc., have been applied to obtain a predictive pattern classifier that is able to detect/predict the pilot



state. A computational and empirical cognitive fatigue and drowsiness model will be developed in order to analyse different signals from aircraft measures. For assessing predictive parameters for cognitive distraction during driving, we run several experiments that produce a relevant amount of data. Working on these offline data, a pattern classifier will be trained and will be able to detect these situations in time. A tool will be developed to wrap this pattern classifier, and will be integrated in the RTP.

### 3.5.6.2 State-of-the-Art

The assessment of mental workload and the identification of the “functional state” of the pilot’s brain could help both in optimising driving tasks (if possible) and planning work–rest rhythms in order to avoid hazardous errors during driving. In addition, such assessment could avoid the occurrence of sustained periods of mental overload during driving, which eventually may result in a drastic decrease of performance with possibly dangerous consequences [66].

It has been already demonstrated by several studies that **EEG** is sensitive to fluctuations in vigilance and has been shown to predict performance degradation due to sustained mental work [121][53]. Associations between a decrease of human alertness and a reduction in vigilance as well as fatigue have been found to generate precise signs in the on-going EEG, especially in alpha and theta waves [148][120][119].

It has now largely been accepted that the increment of the task difficulty leads to an increase of the **heart rate**. The cardiovascular response can thus be used to evaluate the mental load of a task in aviation [57]. However, the variation of heart rate is also linked to different factors besides mental workload, including the fatigue of the subjects (in terms of muscular efforts).

It is a common experience that when a particular task involves the use of visual attention, the subject becomes more concentrated and decreases the time spent with the eye closed for blinking, i.e., their blink frequency decreases. Researchers have investigated whether such phenomena could lead to valid indications about the mental workload for tasks requiring high visual attention, such as driving. As a result, eye blink data has been collected in highly realistic settings of driving. Different parameters (**EOG**) characterising the blink, such as the Blink Rate (BR), the Blink Duration (BD), and the Blink Latency (BL) have been analysed and used as workload measures in a series of studies [42][177][93][157][176].

In [52], [132] generated a sleepiness “detector” based on the ratio between delta and alpha EEG power spectra estimated during different baseline and drowsiness



periods in normal volunteers. Another offline classification approach was proposed by [99], here the authors developed an algorithm that was able to detect several progressive stages of “mental” fatigue as validated by an independent examination of driver’s videos. An artificial neural network (ANN) was applied as a classifier to generate decisions between drowsy and alert periods based on EEG rhythm variations [175], [167]. Increasing the complexity of the classification algorithms, by using support vector machine (SVM) methodologies, allowed a decrease in the number of required electrodes from the 64 [51] to only 19 (plus an EOG channel) while still maintaining a high classification accuracy (99%). By applying a particular SVM classifier, it was possible to significantly correlate the above-mentioned spectral EEG parameters with the occurrence of “high” or “low” reactivity period of pilots during driving. In another study on a large sample of aircraft pilots, a particular neurophysiologic index, which was built on theta and alpha frequency increase/decrease ratios with respect to a baseline condition, was demonstrated to correlate with the pilot’s reports on the difficulty of the task performance [24].

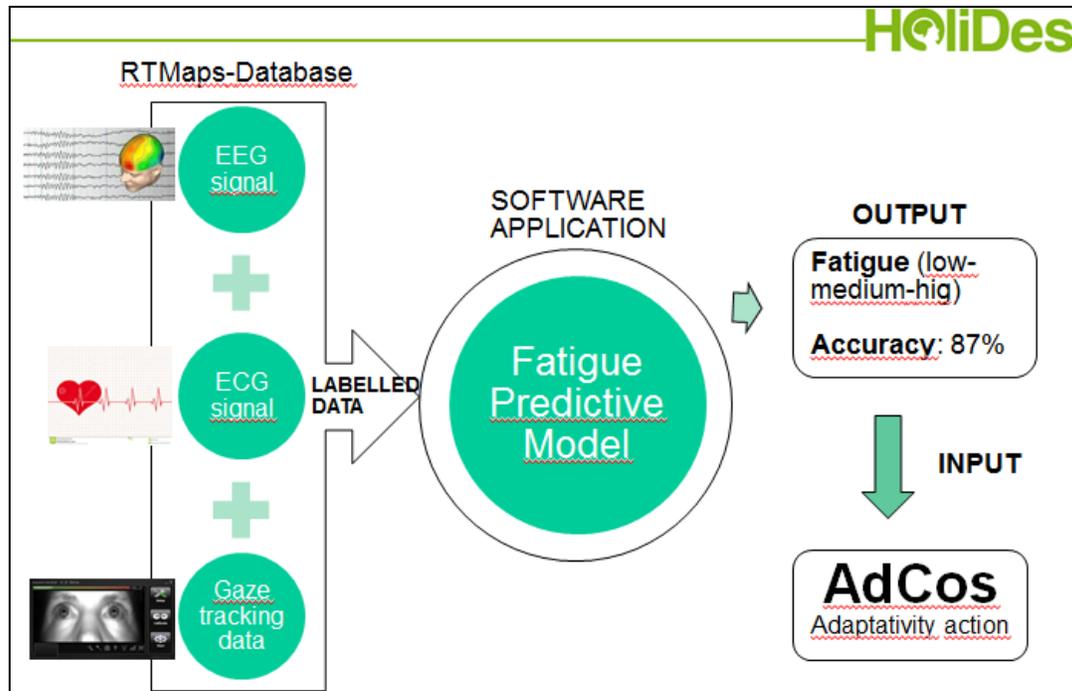
### 3.5.6.3 MTT Description

A tool will be developed to wrap the pattern classifier, and it will be integrated in the RTP. The objective of this tool is to extract information about the workload of the pilot from offline recorded data.

Depending on the workload makers extracted from **input** data (EEG, gaze tracking), several machine learning techniques will be tested to infer the workload of the pilot and obtaining the best scores:

- *Feature Extraction*: the technique aims at building relevant features for classification, using the initial set of attributes (all of them) as the basis of the process.
- *Ensembles*: the technique combines several classification algorithms to obtain the best result. Also known as “Bagging” and “Boosting”.
- *Feature Selection*: the technique aims at extracting the most relevant features for classification, using a genetic algorithm that boosts the whole process
- *ELM-Extreme Learning Machine*: ELM has already been successfully used in Biometrics, Bioinformatics Signal processing, Human action recognition, etc.

Some of these techniques (and probably others) will be tested on our input data to know which technique fits better and provides the best classification results. The **output** of the tool will be the workload of the pilot (a prediction) and the accuracy of this prediction. For instance: PILOT STATE→ workload / not workload (87% of accuracy).



**Figure 54: Predictive software scheme**

### 3.5.7 Driver Distraction Classifier (UTO)

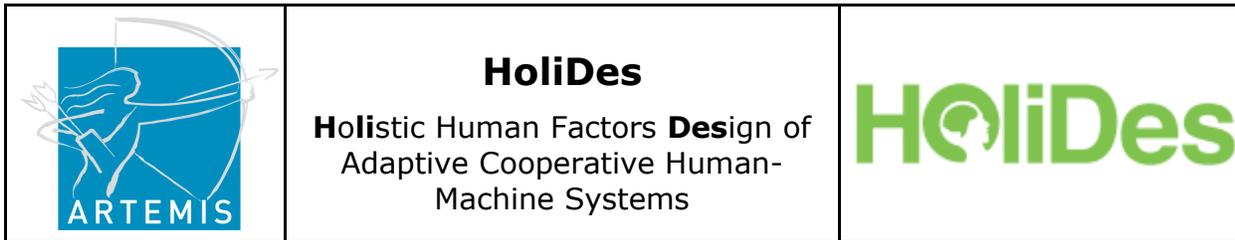
#### 3.5.7.1 Introduction

Attention refers to the process whereby a person concentrates on some specific features of the environment (for a driver it can be external or internal to the vehicle), with the (partial) exclusion of others.

Attention is therefore a multifaceted phenomenon for which many definitions can be found in literature. Anyway, it implies some characteristics:

- Attention is limited, that is we can only attend one thing at a time.
- Attention is selective, that is we can direct our attention to one thing or to something else.
- Attention is linked to consciousness, which is what we are aware of, at any time.

To sum up, attention is a conscious or non-conscious engagement in perceptual, cognitive and / or motor activities. Under this point of view, it is a cognitive activity,



because it refers to mental representation, namely the knowledge a driver has of the world and of him-/her-self.

Actually, it is very difficult to evaluate “an amount of attention” a driver uses or can use; therefore, we focus on the recognition and classification of the inattention (i.e. distraction) which can be assessed by the driver’s observation and behaviour, with a particular interest for the visual type. This means that, for us, distraction is regarded as “the diversion of attention away from activities critical for safe driving towards a competing activity”.

There is accumulating evidence that driver’s distraction is a leading cause of vehicle crashes and incidents. The National Highway Traffic Safety Administration estimates that, in 25% of all crashes, some form of inattention is involved [169].

This is also true at European level, where driver’s distraction contributes to 22% of car crashes and near-crashes [90], and 76% of truck crash and 46% of near crashes [133], as reported from naturalistic driving studies. Detection of drivers’ distraction serves as the fundamental step for an effective distraction countermeasure, such as distraction prevention and mitigation [45], and thus draws increasing attention from multiple research fields.

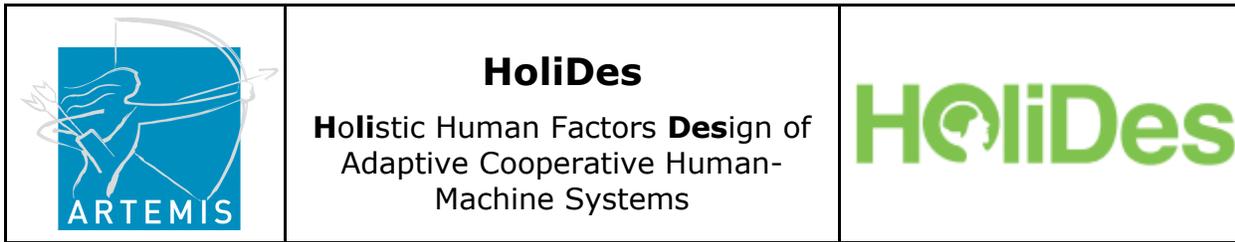
Distraction, as one form of driver inattention, may be characterized as any activity that takes a drivers attention away from the task of driving [45][142].

In particular, it has become an important and growing safety concern with the increasing use of the so-called In-Vehicle Information Systems (IVIS) and Partially Autonomous Driving Assistance Systems (PADAS). Thereby, the detection of the driver status is of paramount importance, in order to adapt IVIS and PADAS accordingly, so avoiding or mitigating their possible negative effects.

Driver's inattention and driver’s distraction do not have a generally accepted definition: the related terms are frequently discussed in the literature, very often they are inconsistently defined and the relationship between them is unclear.

In addition, neither the extent to which driver distraction is responsible for accidents is completely understood. So, it's estimated that 13.3% of crashes involve what they considered distraction and 9.7% were in a category called looked but did not see, [169].

Such a percentage can even increase (+2.6%) if drowsiness is considered as well. The 100-Car Naturalistic Driving Study found that almost 80% of all crashes and 65% of all near-crashes involved driver distraction. In fact, it is well-known that the majority of road accidents (surely > 80%) are due to human error [163], or anyway human (wrong) behaviour, with an increasing evidence that driver distraction and driver inattention are major contributing factors in car and truck crashes and



incidents, with the National Highway Traffic Safety Administration (NHTSA) estimating that, in 25% of all crashes, some form of inattention (including distraction) is involved [182].

Even though the ambiguity in its definition and actual impact, it seems that the scientific community agrees on one thing: driver distraction and inattention is an important safety concern. All in all, drivers distraction is not a new problem in road safety: we may say that it has been around for as long as people have been driving cars.

It is likely that the problem will increase as more wireless or mobile technologies find their way into vehicles. Although in the last few years many European countries have prohibited the use of for example mobile phones when driving, nonetheless it should not be expected that the amount of driving distraction will necessarily decrease. In fact, even without the distraction caused by mobile devices, the use of the so-called In-Vehicle Information Systems (IVIS) e.g. navigation systems can be additional sources of potential distraction. One method, followed by many car-manufacturers and automotive suppliers, aims at minimizing the risk of crashes rather than distraction by means of the development of dedicated supporting systems: the so-called Advanced Driving Assistance Systems (ADAS) and Partially Autonomous Driving Assistance Systems (PADAS), such as lane-keeping assistance system, forward collision warning system, emergency braking system, etc. [179].

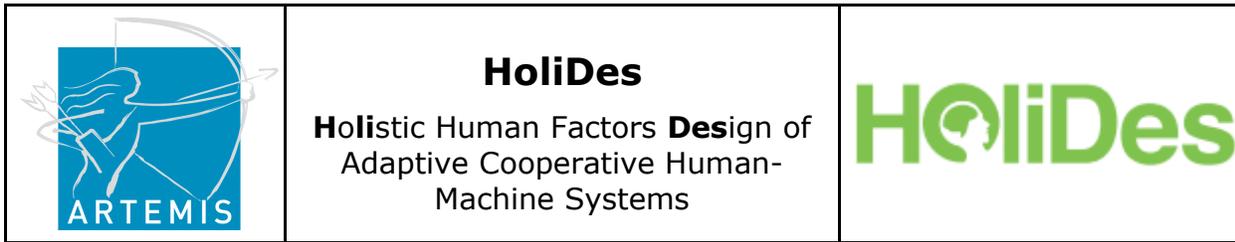
However, it is also true that such PADAS may induce themselves some forms of distraction. In this context, allowing drivers to take benefits from the use of these IVIS and PADAS without diminishing safety is a big and important challenge.

One promising strategy to deal with such a problem involves the classification of drivers status distracted driver, in this case in real time and then using this classification for a twofold goal:

- the adaption of IVIS technologies, in order to mitigate the effects of distraction
- the adaption of PADAS strategies, in order to minimize the effects of distraction on the driving task.

For instance, the intervention of a forward collision assistance system can be triggered, based on the driver state: if distraction is detected the function strategies can be adjusted accordingly (e.g. braking is modulated differently or warning signals are anticipated). On the contrary, if the system detects that the driver is not distracted, but intended to overtake, the warning can be delayed or suppressed, even in case of approaching the vehicle ahead.

Such a smart assistance, which recognizes driver's intention and state, would allow for a greater safety margin, without irritating the driver with false alarms or



inappropriate interventions in normal driving conditions, so enhancing the user acceptability. However, in literature, there is not a unique and commonly agreed definition of distraction, but several ones very often overlapped and mixed with inattention or with other drivers states, such as drowsiness and workload. In particular, we start from the following definition [146], [100]:

*Driver distraction is the diversion of attention away from activities critical for safe driving toward a competing activity.*

This has been extended by Regan et al. [146], adding the concept of Driver Inattention, which means insufficient or no attention to critical activities for safe driving toward a competing activity.

It is worth to note that such a definition suffers from hindsight bias, since it is really difficult to say if the driver is distracted until after something dangerous happens and then it will be too late for the system to intervene.

Nevertheless it would almost be impossible to use the concept of distraction without some preliminary assumptions; even if the situation does not bring to an accident 100 times but it does on the 101st time even though the behaviour is not different however these are potentially critical situations and we want that our systems can prevent such risky conditions (because we don't know which ones can lead to an accident). In fact, in these situations, drivers are not ready to react appropriately to any unexpected event and thus the accidents are more likely.

To sum up, distinct from other forms of driver inattention, distraction occurs when a driver's attention is diverted away from driving by a secondary task that requires focusing on an object, event, or person not related to the driving task.

Although existing data is inadequate and not representative of the driving population, it is estimated that drivers engage in potentially distracting secondary tasks approximately 30% of the time their vehicles are in motion (conversation with passengers is the most frequent secondary task followed by eating, smoking, manipulating controls, reaching inside the vehicle, and cell phone use.). Accordingly to that, we have considered visual distraction as the diversion of visual attention away from the road.

### **3.5.7.2 State-of-the-Art**

First of all, as aforementioned, there are mainly three dimensions of distraction, namely physical, cognitive, and visual distraction. Most existing research studies each type of distraction separately. For example, in the project HASTE [137], each participant experiences one type of distraction (visual/cognitive) at one time, and



different behaviour pattern were observed under different types of distraction. Liang et al [110] focuses on detecting cognitive distraction, which is simulated by an audio mental secondary task.

This is basically also our approach, even if – for the experimental procedure we have followed – in our case cognitive and visual distractions are a little bit mixed together.

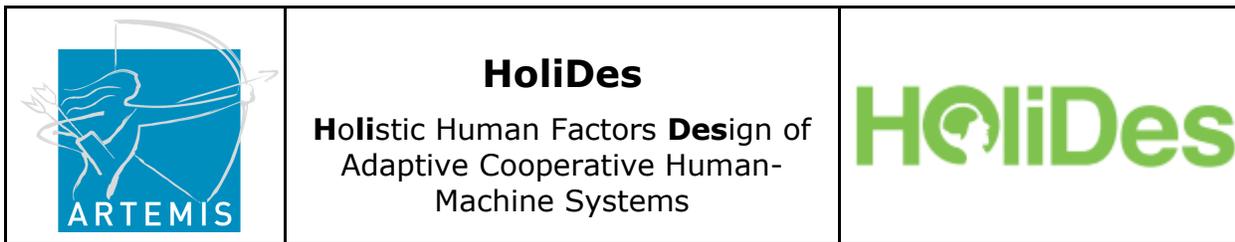
The idea to use Machine Learning (ML) techniques to detect driver's distraction is not completely new. In particular, [179] and [185] suggest that there are basically three approaches to such a recognition problem: monitoring driver's perception; monitoring driver's steering and lane keeping behaviour; recognizing driver's involvement in a given secondary task.

Despite the fact that different classification methods can be found in literature to detect distraction or inattention while driving, nevertheless, since the mental state of the driver is not directly observable, no simple measure can index distraction precisely and thereby all traditional methods show some limits [97]. In this context, the predominant approach is to use ML techniques, which seem to be much appropriated for this type of classification problem.

From a more "philosophical" point of view, one of the most ambitious goals of automatic learning systems is to mimic the learning capability of humans and humans' capability of driving is widely based on experience, particularly on the possibility to learn from experience. From a more technical point of view, data collected from vehicle dynamics and external environment are definitely non-linear. From literature, several studies have proved that in such situations machine learning approaches can outperform the traditional analytical methods. Moreover, also human's driver mental and physical behaviour is non-deterministic (see [37], [35], [79], and [60]).

On the other hand, vehicle dynamics data are user, road and situation dependent and therefore the classifiers, based on ML techniques, are strongly tailored to the conditions and situation that are selected for the training phase. In fact, we suggest building a specific model for each driver, and for each situation. How to adapt and generalize such a model to other situations is still an open problem worth to be investigated.

In our opinion, the most representative works are [46], [179], [108], [104], [97] and [162], since more strongly related to our research and they have been a source of inspiration for us.

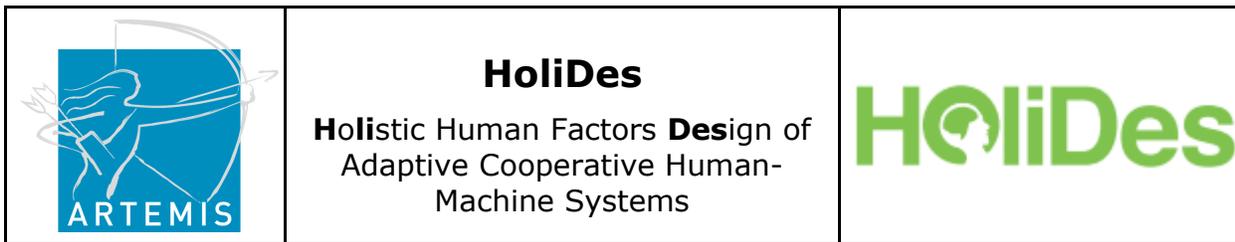


In particular, the predominant approach is to use static classifiers such as support vector machines (SVMs). Liang et al., developed real-time methods for distraction classification using Support Vector Machines [108] and Bayesian Networks [104]. Their results are comparable to ours, since in [108] they achieved a best performance of more than 95%, while in [105], modelling the dynamic of driver's behaviour by using a Dynamic Bayesian Network (DBN), led to accuracies of about 80.1% on average. However, here, the authors pointed out that time dependencies are highly relevant when predicting the current state of a driver. Our best case was > 96%, so – considering also the differences in the experiments, even if both carried out in a driving simulator – absolutely comparable with their best result of 95%. By the way, it is worth noting here that such comparisons can only be indicative, since the datasets are different for each case and also the methods and the tools used for training are not the same.

Similar approaches toward driver behaviour or driver state estimation that model contextual information via DBNs or Markov models can also be found in [139] and [95]. Another promising approach can be found in [97], where SVMs are used to detect driver distraction based on data captured under real traffic conditions, resulting in accuracies of 65%–80%. Features are thereby computed from fixed-length time windows, i.e., the amount of context that is incorporated into the classification decision, is predefined. Other classification strategies include the application of fuzzy logic or neural networks ([159], [184] and [185]).

In addition, it is worth to mention here two specific and recent works: in the former [46], Ersal et al., propose a framework to study the individual effects of secondary tasks and classify driving behaviour. They illustrate that the different effects of secondary tasks on different drivers can be studied using a model-based approach. Furthermore, they point out that using the model-based framework in conjunction with SVMs helps systematically classify driving behaviour as distracted or non-distracted. In details, this SVM classifier is used with a radial-basis neural-network-based modelling framework, developed to characterize the normal driving behaviour of a driver when driving without secondary tasks. Such a developed model is then used in a scenario of driving with a secondary task to predict the hypothetical actions of the driver: the difference between the predicted normal behaviour and the actual distracted behaviour gives individual insight into how the secondary tasks affect the driver. When this framework is used together with SVM, it can help systematically classify normal and distracted driving conditions for each driver.

So, what is really interesting here is that authors consider a model-based approach, where eye-tracker or gaze data are not present; however, in order to build the



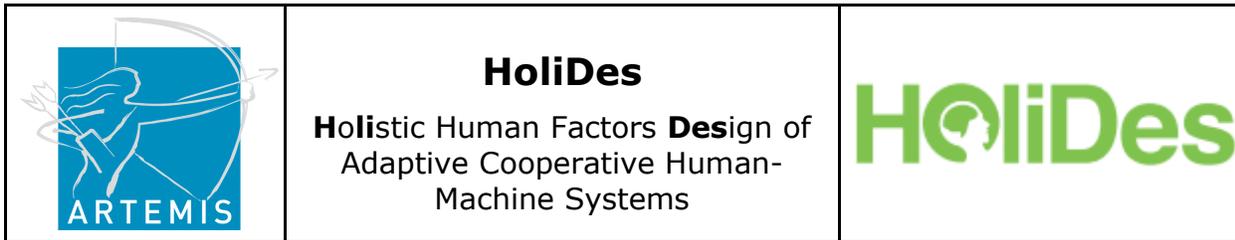
target-set, they state: “for the purposes of the classification, all the instances in normal driving are labelled as vigilant and all the instances in driving with secondary task are labelled as distracted”. This seems to be inadequate to our scenarios.

In the second [179], Wöller et al., introduce a framework and a technique for online driver distraction detection based on modelling contextual information in driving and head tracking data captured during test drives in real traffic. Their approach is based on long short-term memory (LSTM) Recurrent Neural Networks (RNN), exploiting their ability to capture the long-range temporal evolution of data sequences, in order to reliably detect inattention and can be seen as a basis for adaptive lane-keeping assistance. The amount of contextual information that is used for classification is thereby learned by the LSTM network itself during the training phase. This LSTM recurrent neural networks enable a reliable subject-independent detection of inattention with an accuracy > 95%. Thereby, they claim that LSTM framework significantly outperforms conventional approaches such as support vector machines (SVMs).

Most of the aforementioned works used eye-tracker information as inputs to the classifier. When using the simulator, it is relatively easy to have eye-tracker data, but in a real-time application in the car, this is extremely difficult, since there are several limitations. The first concerns the problem of integration: a dedicated camera and related ECU is needed and has to be integrated into the cockpit of the vehicle (with the associated problems of design and costs). Second, although the information provided by eye-tracker device are absolutely useful, nonetheless they require – for example – that the drivers do not wear sunglasses or glasses, or eye make-up, because these conditions may negatively affect tracking accuracy [108]. Moreover, there is the problem to obtain consistent and reliable sensor data. Eye trackers may lose tracking accuracy when vehicles are traveling on rough roads or when the lighting conditions are variable. Of course, the use of other physiological measures (such as heart rate or respiration rate, skin conductance, etc.) can provide other excellent indicators, but they are even more intrusive and difficult to use in real-time in the ordinary cars.

### **3.5.7.3 MTT Description**

We focused our attention toward the neural network paradigm in order to build a suitable tool for driver distraction detection. In this section we provide a detailed description of the structure and the mathematical principles behind the classifier tool, while *in Deliverable 5.4* we will explain how this tool is tailored to the problem of driver distraction.



Neural networks are computational models made up of an interconnected group of simple units, called neurons, which processes information coming from the external environment to identify complex relationships and provide consistent output signals. They are used in various disciplines such as neuroscience, mathematics, statistics, physics or engineering to solve real problems of classification, regression, diagnosis, clustering, control, automation, etc.

Single Layer Feedforward Neural Networks (SLFN) training was mainly accomplished by iterative algorithms involving the repetition of learning steps aimed at minimising the error function, over the space of network parameters; such methods are slow, computationally expensive and can easily lead to poor local minima.

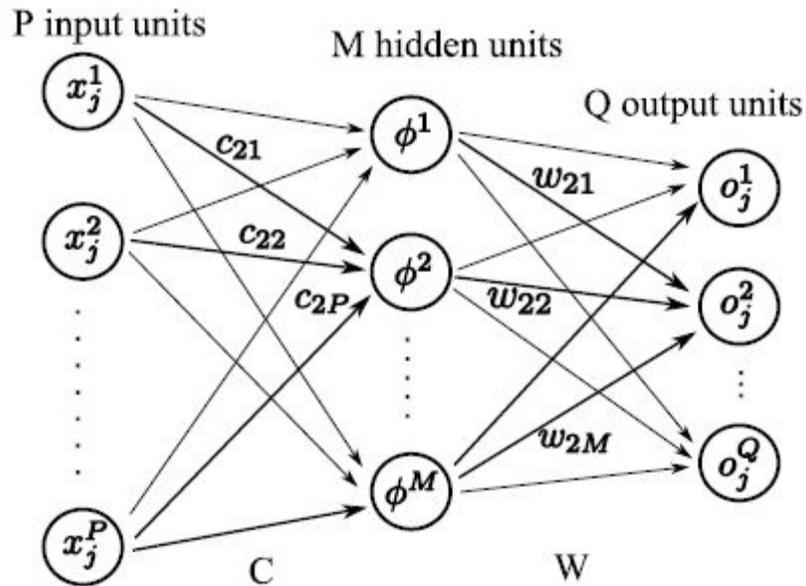
Recently some new techniques based on matrix inversion have been developed, becoming the basis of a complete and exhaustive machine learning theory with the work by Huang and colleagues [73]. Their results state that SLFNs with randomly chosen input weights and hidden layer biases can learn distinct observations with a desired precision, provided that activation functions in the hidden layer are infinitely differentiable.

Besides, output weights are determined by Moore-Penrose generalised inverse (or pseudo-inverse) of the hidden layer output matrix, so iterative training is no more required.

Extreme Learning Machine (ELM) is an algorithm characterized by the fact that input weights are randomly assigned, while output weights are computed using the analytical procedure of pseudo-inversion.

With this method the training reaches the result in one step: ELM can find the minimum training error without using an iterative procedure, notably reducing the computational costs and with good generalization. The only parameter that needs to be kept under control is the choice of input weights during the first phase: error, in fact, depends on this random choice and therefore more attempts are required to reach a good result.

The following Figure 55 shows a standard SLFN with  $P$  input neurons,  $M$  hidden neurons and  $Q$  output neurons, non-linear activation functions  $\Phi$  in the hidden layer and linear activation functions in the output layer.



**Figure 55: A Single Layer Feedforward Neural Network**

Considering a dataset of  $N$  distinct training samples of (input, output) pairs  $(x_j ; t_j)$  where  $x_j \in \mathfrak{R}^P$  and  $t_j \in \mathfrak{R}^Q$  the learning process for a SLFN aims at producing the matrix of desired outputs  $T \in \mathfrak{R}^{N \times Q}$  when the matrix of all input instances  $X \in \mathfrak{R}^{N \times P}$  is presented as input.

As said before in the pseudoinverse approach input weights  $C$  and hidden layer biases are randomly chosen and no longer modified. Thus, output weights  $\omega$  are the solution of the linear system

$$H\omega = T$$

where  $H \in \mathfrak{R}^{N \times M}$  is the hidden layer output matrix of the neural network and it is completely determined after having fixed input weights.

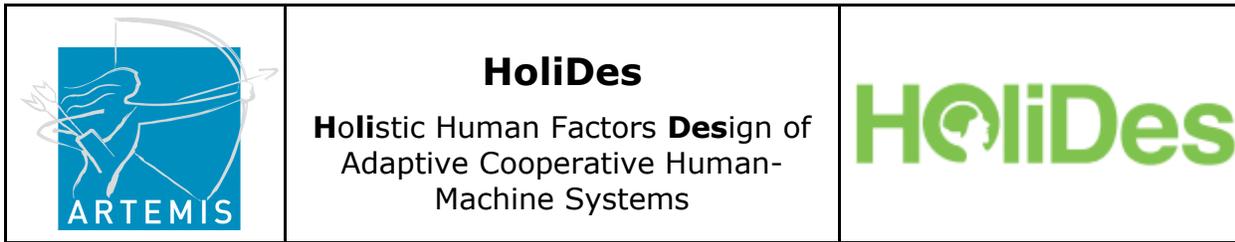
Since  $H$  is a non-invertible matrix, a least square solution  $\omega^*$  has therefore to be searched to minimise the cost functional:

$$E_D = ||H\omega^* - T||^2$$

and it has the expression

$$\omega^* = H^+T$$

where  $H^+$  is the Moore-Penrose generalised inverse (or pseudoinverse) of matrix  $H$ .



Singular value decomposition is a computationally simple and accurate way to compute the pseudoinverse, so it is then used to evaluate the pseudoinverse matrix  $H^+$ .

**ELM learning algorithm:**

Step 1 : Assign arbitrary input weight  $C$  and hidden layer bias.

Step 2 : Calculate the hidden layer output matrix  $H$ . This matrix has a dimension  $N \times M$ , and elements  $h_{l,m} = \Phi(x_l \bullet c_m)$

$$H = \begin{bmatrix} \Phi(x_1 \bullet c_1) & \dots & \Phi(x_1 \bullet c_M) \\ \dots & \dots & \dots \\ \Phi(x_N \bullet c_1) & \dots & \Phi(x_N \bullet c_M) \end{bmatrix}$$

Step 3 : Calculate the output weight  $\omega$ , solution of the following system

$$H\omega = T \Rightarrow \omega^* = H^+T$$

where  $T$  indicates the Target vector, built with the desired responses as output.

The following is a list of some of the main features of the ELM algorithm:

- ELM is extremely faster compared with the classical learning algorithm.
- ELM tends to reach not only the smallest training error but also the smaller norm of weights so it has better generalization performance.
- Unlike the traditional classic method, the ELM learning algorithm looks simpler.

While Back-Propagation algorithm can be used for feedforward neural networks which have more than one hidden layer, the ELM algorithm, is only valid for single hidden layer feedforward networks(SLFNs).

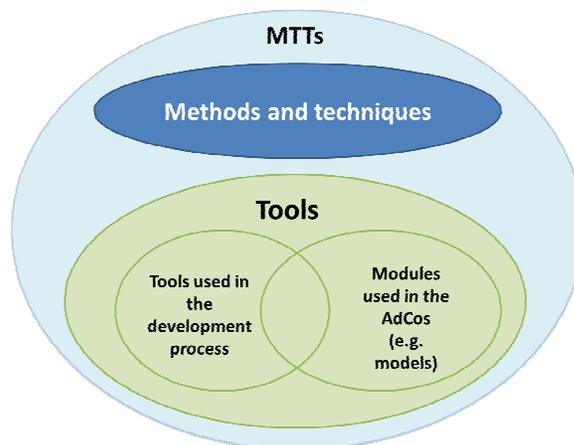
## 4 Integration Plans

The integration of MTTs into the HF-RTP will be based on a detailed plan and methodology defined in D1.5 section 2.4.4 HF-RTP integration methodology.

In particular, the HF-RTP will extend the CESAR RTP with human factors engineering activities and will integrate the HoliDes techniques and tools by defining RTP workflows and processes, a Common Meta Model and an interoperability standard tailored on the needs of the Human Factors.

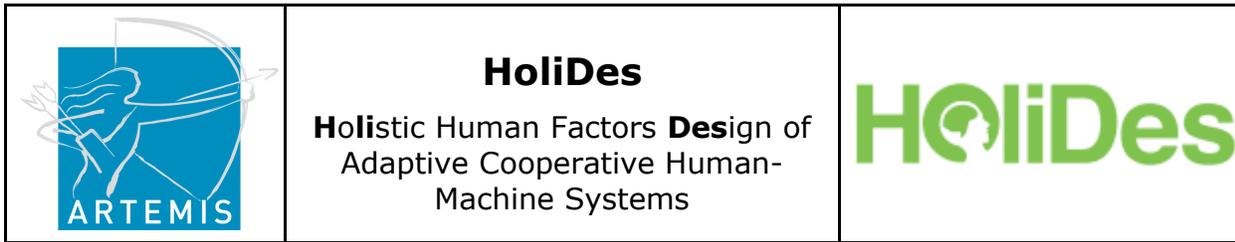
As regards the tools, two different categories of software have been developed (as shown in Figure 56):

- The tools that are meant to be integrated into the tool chains of the AdCoS owners to improve their development process (e.g. Magic-PED or OFF for the task modelling)
- The tools that have been developed to be embedded into the AdCoS to improve its functionalities (e.g. the Driver Distraction Classifier of TWT)



**Figure 56: MTTs – methods, techniques and tools**

Moreover, some of them present a double nature: according to the context of use, they can be either used in the development process or “as bricks” of an AdCoS (e.g. a module for the detection of the distraction can be included in the AdCoS to improve its functionalities and it can be employed for the evaluation of an AdCoS to assess if it induces distraction in the operator).



For the tools, the integration implies the development of specific piece of software (i.e. adapters) to implement the OSLC specification and allow the sharing of data with other tools.

The overall methodology for the tools and modules foresees the following activities to be integrated into the HF-RTP:

- 1) Identify their inputs and outputs, related MTTs or AdCoS. Moreover, only for tool, identify their compliance with OSCL and the estimated date for integration into HF-RTP.
- 2) Identify the correspondence between its inputs and outputs and the concepts of the HoliDes Meta Model.
- 3) Only for tools: develop the adapters or parsers between the concepts and the data managed by the MTT (parse numerical values and strings into RDF and from RDF to numerical values and strings).

In this particular deliverable, we will focus on Step 1 of the described methodology. The work done so far for the integration plan will be described in the Annex III HoliDes Integration Plan.

## 5 Requirements Update

The analysis process has been described in D2.1. In total, 440 requirements from the application work packages have been analysed in the first cycle, and now been updated. The list of requirements consisted of requirements dedicated to the development of the AdCoS, and for the RTP. In addition to the RTP requirements, the AdCoS requirements have been analysed, too, because it may be the case that within the AdCoS requirements, relevant information can be found for the development of the WP2 models and the developed MTTs.

Annex I shows the assignment of the requirements to the tools for the second cycle. In addition to the update of requirements, the list of tools has been updated. The following MTTs have been previously assigned to WP2, but now been removed:

MTT	Reason for WP2 removal
HMFDIM (IFS)	Developed in other WP
Tobii glasses (SNV)	3rd Party tool not developed by partner
FaceLab 5 + Eyeworks (SNV)	3rd Party tool not developed by partner
Captiv T-sens (SNV)	3rd Party tool not developed by partner
Enobio (SNV)	3rd Party tool not developed by partner
HS-Searchopt	Replaced by other tool in WP7

Table 6 gives an overview on the current status:

Status	Total	not relevant	need feedback	assigned	accepted	rejected	in progress	in test	fulfilled
RTP req.:	184	96	8	28	5	9	34	9	5
AdCoS req.:	252	201	3	42	1	8	2	1	1
<b>Total</b>	<b>436</b>	297	11	70	6	17	36	10	6

**Table 6: Overview of current requirements status**

In the given table, a requirement has only been counted once as e.g. assigned, also in case more than one MTT has assigned the status.

Below are indicated the evolution of the requirements and of their assignation to MMT tools for the different domains.

In **Annex I: Requirements Update**, the colours in the table have the following signification:

- Red: Requirement or assignment rejected
- Green: Requirement or assignment accepted
- Yellow: Change in names/description/version of the requirement
- Grey: Evolution of status

## 5.1 Changes in the status of the deliverable for WP6

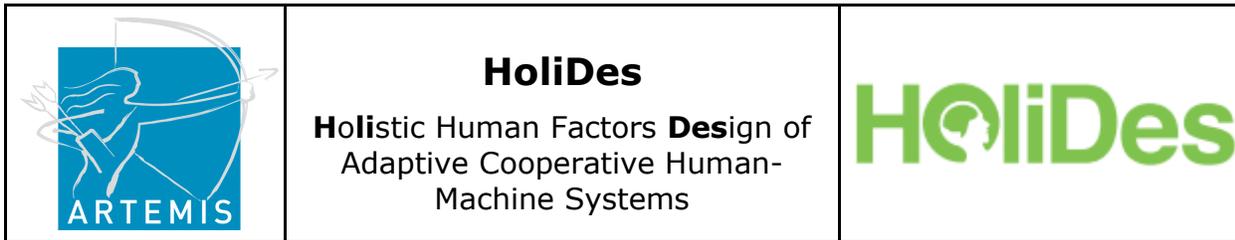
### New requirements:

No requirement added

### Deleted requirements

2 requirements rejected:

- WP6\_PHI\_HEA\_REQ13
- WP6\_AWI\_HEA\_REQ01



### **Changes in Names and/or Description**

2 requirements modified:

- WP6\_AWI\_HEA\_REQ03
- WP6\_PHI\_HEA\_REQ14

### **Evolution of the status of tool assignment**

No evolution in the status

## **5.2 Changes in the status of the deliverable for WP7**

### **New requirements**

13 new requirements have been added:

- WP7\_TRS\_AER\_REQ\_36
- WP7\_TRS\_AER\_REQ\_37
- WP7\_TRS\_AER\_REQ\_38
- WP7\_TRS\_AER\_REQ\_39
- WP7\_TRS\_AER\_REQ\_40
- WP7\_HON\_AER\_REQ87
- WP7\_HON\_AER\_REQ88
- WP7\_HON\_AER\_REQ89
- WP7\_HON\_AER\_REQ90
- WP7\_HON\_AER\_REQ91
- WP7\_HON\_AER\_REQ92
- WP7\_HON\_AER\_REQ93
- WP7\_HON\_AER\_REQ94

### **Deleted requirements**

No requirement rejected

### **Evolution of the status of tool assignment**

No evolution in the status

## **5.3 Changes in the status of the deliverable for WP8**

### **New requirements**

No requirement added

### **Deleted requirements**

7 requirements rejected:



## HoliDes

Holistic Human Factors Design of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- WP8\_IRN\_CR\_REQ03,
- WP8\_IRN\_CR\_REQ04
- WP8\_IRN\_CR\_REQ10
- WP8\_IRN\_CR\_REQ12
- WP8\_IRN\_CR\_REQ13
- WP8\_IRN\_CR\_REQ15
- WP8\_IRN\_CR\_REQ18

### Evolution of the status of tool assignment

- WP8\_ADS\_CTR\_REQ05
- WP8\_ADS\_CTR\_REQ06
- WP8\_ADS\_CTR\_REQ12
- WP8\_ADS\_CTR\_REQ13
- WP8\_ADS\_CTR\_REQ17
- WP8\_ADS\_CTR\_REQ18
- WP8\_ADS\_CTR\_REQ24
- WP8\_ADS\_CTR\_REQ25
- WP8\_ADS\_CTR\_REQ26
- WP8\_ADS\_CTR\_REQ27
- WP8\_IRN\_CR\_REQ01
- WP8\_IRN\_CR\_REQ02
- WP8\_IRN\_CR\_REQ05

## 5.4 Changes in the status of the deliverable for WP9

### New requirements:

No requirement added

### Deleted requirements

No requirements rejected

### Evolution of the status of tool assignment

- WP9\_IFS\_AUT\_REQ03
- WP9\_IFS\_AUT\_REQ04
- WP9\_IFS\_AUT\_REQ05
- WP9\_IFS\_AUT\_REQ06
- WP9\_IFS\_AUT\_REQ07
- WP9\_IFS\_AUT\_REQ10
- WP9\_IFS\_AUT\_REQ11
- WP9\_TWT\_AUT\_REQ01
- WP9\_TWT\_AUT\_REQ03



## HoliDes

**H**olistic Human Factors **D**esign of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- WP9\_TWT\_AUT\_REQ04
- WP9\_TWT\_AUT\_REQ06
- WP9\_OFF\_AUT\_REQ03
- WP9\_OFF\_AUT\_REQ05
- WP9\_OFF\_AUT\_REQ06

### Changes in Names and/or Description

- WP9\_OFF\_AUT\_REQ06



## 6 Summary

Objective of WP2 is to develop modelling languages that support the modelling of adaptive and cooperative Systems (AdCoS), as well as editors for the specification of these models.

The development of the modelling languages has been started:

- HMI Interaction, Training, and Resource Modelling languages have been formalized and improved
- Initial version of task model discussed and further task models have been collected, integration into common task model ongoing
- Classifications of Human Operator Models started.
- Formalizations of the MDP/MDPN and the DBN languages have been continued as behavioural operator models
- Work on cooperation model has been started

In addition to that, several tools have been developed and are available to the partners for applying them in the development of their AdCoS. The new tools described in this deliverable are Djnn, Pilot Pattern Classifier and Driver Distraction classifier.

A new section about Integration has been included in this deliverable. This section is the result of applying the methodology described in *D1.5*. The first steps of this methodology have been applied to each one of the MTTs assigned to the WP2 and related to modelling.

There is a clear progress in the requirements visible, in the initial requirements analysis, with their indicated evolution and their assignation to MTT for the different domains. Also some issues on the requirements that needed feedback have been resolved. Some of them have been rejected because of this feedback, while some are assigned.



## 7 References

- [1] Aasman, J. (1995). Modeling driver behaviour in SOAR. In Leidschendam, The Netherlands: KPN Research.
- [2] Agamennone, G., Nieto, J. I., Nebot, E. M. (2011). A Bayesian Approach For Driving Behavior Inference. In 2011 IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, June 5-9, 2011, pp. 595-600, IEEE Catalog Number: CFP11IVS-CDR, ISBN: 978-1-4577-0889-3.
- [3] Ahmed, K., I. (1999). Modeling Driver's Acceleration and Lane Changing Behavior. PhD thesis, Massachusetts Institute of Technology.
- [4] Ajzen, I. (2002): Perceived Behavioral Control, Self-Efficacy, Locus of Control, and the Theory of Planned Behavior. In Journal of Applied Social Psychology, 32, pp. 665-683.
- [5] Alfaro, L. de: Stochastic Transition Systems. In Proc. of 9th International Conference on Concurrency Theory (CONCUR'98), volume 1466 of LNCS, pages 423-438. Springer, 1998.
- [6] Anderson, J.R. (2000). Learning and Memory: An Integrated Approach. 2nd Edition, Wiley.
- [7] Anderson, J.R. and Lebiere, C.J.: The Atomic Components of Thought. Lawrence Erlbaum Associates, June 1998.
- [8] Anderson, J.R.: How can the Human Mind Occur in the Physical Universe. Oxford Series on Cognitive Models and Architectures. Oxford University Press, August 2009.
- [9] Annett, J.: Hierarchical Task Analysis. In: E. Hollnagel (ed). Handbook of Cognitive Task Design. Lawrence Erlbaum Associates, Mahwah, New Jersey, 17-35, 2003
- [10] Aoude, G. S., Desaraju, V. R., Stephens, L. H., How, J. P. (2011). Behavior Classification Algorithms at Intersections and Validation using Naturalistic Data. In 2011 IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, June 5-9, 2011, pp. 601-606, IEEE Catalog Number: CFP11IVS-CDR, ISBN: 978-1-4577-0889-3.
- [11] Appert, C. and Beaudouin-Lafon, M. (2008). SwingStates: Adding state machines to Java and the Swing toolkit. Journal Software Practice and Experience. 38, 11 (Sep. 2008), 1149-1182.
- [12] Artho, J., Schneider, S., Boss, C.: Unaufmerksamkeit und Ablenkung: Was macht der Mensch am Steuer? - Transport Research International Documentation - TRID. Online: <http://trid.trb.org/view.aspx?id=1244037>, last access 21.05.2014
- [13] Bando, T., Miyahara, T., Tamatsu, Y. (2011): Traffic Interactions: Estimate Driving Behavior's Influence. In 2011 IEEE Intelligent Vehicles Symposium



## HoliDes

Holistic Human Factors Design of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- (IV), pp. 546-551, Baden-Baden, Germany, June 5-9, 2011, IEEE Catalog Number: CFP11IVS-CDR, ISBN: 978-1-4577-0889-3.
- [14] Beccuti, M., Franceschinis, G., Haddad, S.: Markov Decision Petri Net and Markov Decision Well-formed Net formalisms. Proc of the 28th Int. Conference on Applications and Theory of Petri Nets and other Models of Concurrency 2007. LNCS vol. 4546:43-62, 2007
- [15] Bellamy, R., John, B. E., und Kogan, S.: Deploying cogtool: Integrating quantitative usability assessment into real-world software development. In Proceedings of 33rd International Conference on Software Engineering (ICSE), S. 691-700, 2011
- [16] Bellet T., Bailly B., Mayenobe P., Georgeon O; Cognitive modelling and computational simulation of drivers mental activities. In P. C. Cacciabue & C. Re (Eds.), Critical Issues in Advanced Automotive Systems and Human-Centred Design, Milan, Springer, pp.317-345, 2007
- [17] Bellet, T., Bailly-Asuni, B., Mayenobe, P., Banet, A.: A theoretical and methodological framework for studying and modelling drivers' mental representations, *Safety Science*, 47, pp. 1205-1221, 2009
- [18] Bellet, T., Mayenobe, P., Bornard, J.C., Gruyer, D., Claverie, B.: A computational model of the car driver interfaced with a simulation platform for future Virtual Human Centred Design applications: COSMO-SIVIC, *Engineering Applications of Artificial Intelligence*, 25, pp. 1488-1504, 2012
- [19] Bellet, T., Mayenobe, P., Bornard, J.C., Gruyer, D., Claverie, B. (2012). A computational model of the car driver interfaced with a simulation platform for future Virtual Human Centred Design applications: COSMO-SIVIC, *Engineering Applications of Artificial Intelligence*, 25, pp. 1488-1504.
- [20] Berndt, H., Emmert, J., Dietmayer, K. (2008). Continuous driver intention recognition with Hidden Markov Models. In Proceedings of the 11th International IEEE on Intelligent Transportation System, pp. 1189-1194.
- [21] Boeing Commercial Airplane. "Airplane Safety. Statistical summary of commercial jet airplane accidents - worldwide operations", Boeing, P.O. Box 3707 M/S 67-TC Seattle, Whashington 98124-2207, 2002.
- [22] Bohnenkamp, H., D'Argenio, P. R., Hermanns, H., Katoen, J.-P.: Modest: A compositional modeling formalism for hard and softly timed systems. *IEEE Transactions on Software Engineering*, 32(10):812-830, 2006.
- [23] Börger, J. (2013). Fahrerintentionserkennung und Kursprädiktion mit erweiterten Maschinellen Lernverfahren. Dissertation, Universität Ulm.
- [24] Borghini, G., Isabella, R., Vecchiato, G., Toppi, J., Astolfi, L., Caltagirone, C., Babiloni, F., 2011. Brainshield HREEG study of perceived pilot mental workload. *Italian Journal of Aerospace Medicine* 5, 34-47.
- [25] Brackstone, M., McDonald, M. (1999). Car-following: a historical review. In *Transportation Research Part F*, 2, pp. 181-196.
- [26] Bruseberg, A. & Shepherd, A.: Job Design in Integrated Mail Processing. In: D. Harris (ed) *Engineering Psychology and Cognitive Ergonomics*. Volume



## HoliDes

Holistic Human Factors Design of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- Two: Job Design and Product Design. Ashgate Publishing, Aldershot, Hampshire, (25-32), 1997.
- [27] Cacciabue, P.C. (ed.) (2007). Modelling Driver Behaviour in Automotive Environments. London: Springer, ISBN-10: 1-84628-617-4.
- [28] Card, S. K., Moran, T. P. & Newell, A.: The Psychology of Human-Computer Interaction. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers, 1983.
- [29] Card, Stuart, Moran, Thomas P., and Newell, Allen, *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ (1983).
- [30] Chatty, S. (1994). Extended a graphical toolkit for two handed interaction. *Proceedings of the ACM conference on User Interface Software and technologies (UIST'94)*. ACM Press, 1994.
- [31] Chatty, S. (2004), Extending a Graphical Toolkit for Two-Handed Interaction. *Proceedings of the 17th ACM symposium on User Interface Software and Technology (UIST 2004)* ACM Press, 2004, pp 267-276.
- [32] Constanine, Larry L. and Lockwood, Lucy A.D., 1999. Software for Use: A Practical Guide to the Models and Methods of Usage-Centred Design, Addison-Wesley ACM Press, New York, ISBN-10: 0321773721
- [33] Cooper, R. and Fox, J.: Cogent: A visual design environment for cognitive modeling. *Behavior Research Methods, Instruments, & Computers*, 30(4):553-564, 1998
- [34] Corker, K.M. and Smith. B.R.: An architecture and model for cognitive engineering simulation analysis: Application to advanced aviation automation. In Proceedings of AIAA Computing in Aerospace 9 Conference, San Diego, CA, 21 October 1993.
- [35] D’Orazio, T., Leo, M., Guaragnella, C., and Distanto, A., 2007. “A visual approach for driver inattention detection,” *Pattern Recognit.*, vol. 40, no. 8, pp. 2341-2355, Aug. 2007.
- [36] D3CoS deliverable 3.08 – Platform for modelling and simulation
- [37] de Waard, D., Brookhuis, K. A., and Hernandez-Gress, N., 2001. “The feasibility of detecting phone-use related driver distraction,” *Int. J. Vehicle Des.*, vol. 26, no. 1, pp. 85-95.
- [38] Doshi, A., Trivendi, M. (2008). A comparative exploration of eye gaze and head motion cues for lane change intent prediction. In Proceedings of the 2008 IEEE Intelligent Vehicles Symposium, pp. 49-54.
- [39] Dragicevic, P. and Fekete, J.-D. (2004). Support for Input Adaptability in the Icon Toolkit. Proceedings of the Sixth International Conference on Multimodal Interfaces (ICMI '04), State College, PA, USA, October 13 - 15, 2004. ACM Press, New York, NY, pp 212-219.
- [40] Dutch Safety Board, 2010. Crashed during approach, Boeing 737-800, near Amsterdam Schiphol Airport, 25 February 2009. Final report of Dutch Safety Board, May 2010.



## HoliDes

**H**olistic Human Factors **D**esign of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- [41] Dzaack-J., Wiesner A., Urbas L. & Heinath, M.: Simplifying the development and the analysis of cognitive models. In Proceedings of EuroCogSci07, Delphi, Greece, 2007.
- [42] Eggemeier, F.T., Biers, D.W., Wickens, C.D., Andre, A.D., Vreuls, D., Billman, E.R., Schueren, J., 1990. Performance assessment and workload evaluation systems: analysis of candidate measures. Technical Report No. HSD-TR-90-023. Brooks Air Force Base, Armstrong Aerospace Medical Research Laboratory, TX.
- [43] Eilers, M., Möbus, C. (2014). Discriminative Learning of Relevant Percepts for a Bayesian Autonomous Driver Model. In Proceedings of the Sixth International Conference on Advanced Cognitive Technologies and Applications (COGNITIVE 2014), IARIA, ISSN: 2308-4197, ISBN: 978-1-61208-340-7.
- [44] Elliott, C. and Hudak (1997), P. Functional reactive animation. In *International Conference on Functional Programming*, pp. 163-173, June 1997.
- [45] Engst Örm, J. and Victor, T. W., "Real-time distraction countermeasures," M. A. Regan, J. D. Lee, and K. L. Young, Eds. Boca Raton, FL, USA: CRC Press Taylor & Francis Group, Oct. 2008, pp. 465-483.
- [46] Ersal T., Fuller, H.J.A., Tsimhoni, O., Stein, J.L and Fathy, H.K., 2010. "Model-based Analysis and Classification of Driver Distraction under Secondary Tasks" in *IEEE Transaction on Intelligent Transportation System* , vol. 11, No. 3, September 2010, IEEE Society Editor.
- [47] Fishbein, M., Ajzen, I. (1975). *Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research*. Reading, MA: Addison-Wesley.
- [48] Forbes, J., Huang, T., Kanazawa, K., Russell, S. (1995). The BATmobile: Towards a Bayesian Automated Taxi. In Proceedings of International Joint Conference on Artificial Intelligence.
- [49] Foyle, C.F. & Hooley, B.L.: *Human Performance Modeling in Aviation*. CRC Press, Taylor & Francis Group, Boca Raton, FL, 2008.
- [50] Freed, M.A.: *Simulating Human Performance in Complex, Dynamic Environments*. Dissertation, Northwestern University, Evanston, Illinois, June 1998
- [51] Fu, J.W., Li, M., Lu, B.L., 2008. Detecting drowsiness in driving simulation based on EEG. In: *Autonomous Systems – Self – Organization Management and Control*, pp. 21-28.
- [52] Gevins, A., Leong, H., Du, R., Smith, M.E., Le, J., DuRousseau, D., 1995. Toward measurement of brain function in operational environments. *Biological Psychiatry* 43, 145-152.
- [53] Gevins, A.S., Bressler, S.L., Cutillo, B.A., Illes, J., Fowler-White, R.M., 1990. Effects of prolonged mental work on functional brain topography. *Electroencephalography and Clinical Neurophysiology* 76, 339-350.



## HoliDes

**H**olistic Human Factors **D**esign of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- [54] Ghahramani, Z., Jordan, M. I. (1997). Factorial Hidden Markov Models. In *Machine Learning*, p.1-31
- [55] Gore, B.F., Hooey, B.L., Wickens, C.D., Socash, C., Gosakan, M., Gacy, M., Brehon, M., und Foyle, D.C.: Workload as a performance shaping factor in MIDAS v5. Proceedings of the Behavioral Representation in Modeling and Simulation (BRIMS) 2011 Conference, 26 March 2011.
- [56] Halbwachs, N. et al. *The Synchronous Data Flow Programming Language LUSTRE*. In Proc. IEEE 1991 Vol. 79, No. 9.
- [57] Hanson, E.K.S., Bazanski, J., 2001. Ecological momentary assessments in aviation: the development of a Pilot Experience Evaluating Device (PEED) for the in-flight registration of flight phases, mental effort, and reaction time. In: Fahrenberg, J., Myrtek, M. (Eds.), *Progress in Ambulatory Assessment Computer Assisted Psychological and Psychophysiological Methods in Monitoring and Field Studies*, pp Measurements. Hogrefe & Huber, Seattle, pp. 477-492.
- [58] Harel, D. (1987). Statecharts: a visual formalism for complex systems. *Science of Computer Programming* (8), pp. 231-274.
- [59] Hartson, H. R. and Gray, P. D.: Temporal aspects of tasks in the user action notation. *Human Computer Interaction* 7, pp. 1-45, 1992
- [60] Healey, J. A. and Picard, R.W., 2005. "Detecting stress during real-world driving tasks using physiological sensors," *IEEE Trans. Intelligent Transp. Syst.*, vol. 6, no. 2, pp. 156-166, Jun. 2005.
- [61] Hoc et al. Evaluation of human-machine cooperation modes in car driving for safe lateral control in curves: delegation and mutual control modes. 2006.
- [62] Hoc, Jean-Michel. Human and automation: a matter of cooperation. Pruski, A. HUMAN 07, 2007, Timimoun, Algeria. Université de Metz, pp.277-285.
- [63] Hoc. Human and Automation: a Matter of Cooperation. In Proceedings of HUMAN-07, 2007.
- [64] Hodgkinson, G.P. & Crawshaw, C.M.: Hierarchical Task Analysis for Ergonomics Research. An Application of the Method to the Design and Evaluation of Sound Mixing Consoles. *Applied Ergonomics* 16 (4) 289-299, 1985
- [65] Hollnagel, E.: *Human Reliability Analysis: Context and Control*. London: Academic Press London, 1993.
- [66] Holm, A., 2010. Developing neurophysiological metrics for the assessment of men-tal workload and the functional state of the brain. Doctoral dissertation for the degree of Doctor of Philosophy. Aalto University School of Science and Technol-ogy (Espoo, Finland), 12th of May 2010.
- [67] Horberry, T.; Anderson, J.; Regan, M.A.; Triggs, T.J.; Brown, J.: Driver distraction: the effects of concurrent in-vehicle tasks, road environment complexity and age on driving performance. In: *Accid Anal Prev* 38 (1), S. 185-191. DOI: 10.1016/j.aap.2005.09.007, 2006.



## HoliDes

Holistic Human Factors Design of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- [68] Horberry, Tim; Anderson, Janet; Regan, Michael A.; Triggs, Thomas J.; Brown, John (2006): Driver distraction: the effects of concurrent in-vehicle tasks, road environment complexity and age on driving performance. In: *Accid Anal Prev* 38 (1), S. 185–191. DOI: 10.1016/j.aap.2005.09.007.
- [69] <http://www.media-interactive.de/>
- [70] <http://www.prodefis.de/aviation/de/products/prodefis-course.html>
- [71] <http://www.skymanager.com/>
- [72] <http://www.talon-systems.com/>
- [73] Huang, G.-B., Zhu, Q.-Y. and Siew, C.-K. (2006). Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. In: *Extreme Learning Machine: Theory and Applications* vol.70, pp.489-501.
- [74] International Ergonomics Association. What is Ergonomics. Website. Retrieved 22.06.2015
- [75] Itti, L. and Koch, C. Computational Modeling of Visual Attention, *Nature Reviews Neuroscience* 2(3):194-203 2001
- [76] Itti, L., Koch, C. and Niebur, E.: A. Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(11):1254-1259 1998
- [77] Itti, L., Koch, C.: A saliency-based search mechanism for overt and covert shifts of visual attention, *Vision Research*, Vol. 40, No. 10-12, pp. 1489-1506, May 2000.
- [78] Jenkins, D. P., Stanton, N.A., Walker, G.H., Salmon, P.M.: *Cognitive Work Analysis: Coping with Complexity*, 2012, Ashgate Publishing.
- [79] Ji, Q., Lan, P., and Looney, C., 2006. "A probabilistic framework for modeling and real-time monitoring human fatigue," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 36, no. 5, pp. 862–875, Sep. 2006.
- [80] John, B. E. & Salvucci, D. D.: Multipurpose prototypes for assessing user interfaces in pervasive computing systems. *IEEE Pervasive Computing*, 4(4), pp 27-34, 2005.
- [81] John, B. E. (1990). Extensions of GOMS Analyses to Expert Performance Requiring Perception of Dynamic Visual and Auditory Information. *CHI '90 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.
- [82] John, B. E., Prevas, K., Salvucci, D. D., and Koedinger, K.: Predictive human performance modeling made easy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '04*, pages 455–462, ACM New York, NY, USA, 2004
- [83] John, Bonnie and Kieras, David E., *The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast*, *ACM Transactions on Computer-Human Interaction* 3,4 (December 1996b), 320-351.
- [84] Jonsson, B., Larsen, K. G., and Yi, W.: Probabilistic extensions of process algebras. In *Handbook of Process Algebra*, pages 685–710. Elsevier, 2001.



## HoliDes

**H**olistic Human Factors **D**esign of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- [85] Jordan, M. I., Ghahramani, Z., Saul, I. K. (1996). Hidden Markov Decision Trees. In Proceedings of the 9th Conference on Advances in Neural Information Processing Systems, pp. 01-507.
- [86] Jürgensohn, T. (2007). Control theory models of the driver. In Cacciabue, P. C., (ed.), Modelling Driver Behaviour in Automotive Environments, pp. 277-292, London, Springer.
- [87] Kaplan, K. and Akin, H. L. (2011). Expert System Design for an Autonomous Driver Evaluation System, In 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 314-319, Baden-Baden, Germany, June 5-9, 2011, IEEE Catalog Number: CFP11IVS-CDR, ISBN: 978-1-4577-0889-3.
- [88] Kasper, D., Weidl, G., Dang, T., Breuel, G., Tanke, A., Rosenstiel, W. (2011). Object-Oriented Bayesian Networks for Detection of Lane Change Maneuvers, In 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 673-678, Baden-Baden, Germany, June 5-9, 2011, IEEE Catalog Number: CFP11IVS-CDR, ISBN: 978-1-4577-0889-3.
- [89] Kishimoto, Y., Abe, K., Miyatake, H., Oguri, K. (2008). Modeling Driving Behavior With Dynamic Bayesian Networks and Estimate of Mental State. In Proceedings of the 15<sup>th</sup> World Congress on Intelligent Transport Systems and ITS America's 2008 Annual Meeting.
- [90] Klauer, S. G., Dingus, T. A., Neale, V. L., Sudweeks, J. D., and Ramsey, D. J., "The impact of driver inattention on near-crash/crash risk: An analysis using the 100-car naturalistic driving study data," Tech. Rep., 2006.
- [91] Kobiela, F. (2011). Fahrerintentionserkennung für autonome Notbremsysteme. Springer.
- [92] Koller, D., Friedman, N. (2009). Probabilistic Graphical Models: Principles and Techniques. MIT Press.
- [93] Kramer, A.F., 1991. Physiological metrics of mental workload: a review of recent progress. In: Damos, D.L. (Ed.), Multiple Task Performance. Taylor & Francis, London.
- [94] Kujala, T., & Salvucci, D. D. (2015). Modeling visual sampling on in-car displays: The challenge of predicting safety-critical lapses of control. International Journal of Human-Computer Studies, 79, 66-78.
- [95] Kumagai, T. and Akamatsu, M., "Prediction of human driving behavior using dynamic Bayesian networks," IEICE Trans. Inf. Syst., vol. E89-D, no. 2, pp. 857-860, Feb. 2006.
- [96] Kumagai, T., Akamatsu, M. (2006). Prediction of Human Driving Behavior Using Dynamic Bayesian Networks. In IEEE Trans. Inf. & Syst., Vol. E89-D, No.2.
- [97] Kutila, M. H., Jokela, M., Mäkinen, T., Viitanen, J., Markkula, G., and Victor, T. W., 2007. "Driver cognitive distraction detection: Feature estimation and implementation," Proc. Inst. Mech. Eng., Part D: J. Automobile Eng., vol. 221, no. 9, pp. 1027-1040.



## HoliDes

**H**olistic Human Factors **D**esign of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- [98] Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, Proc. 23rd International Conference on Computer Aided Verification (CAV'11), volume 6806 of LNCS, pages 585–591. Springer, 2011.
- [99] Lal, S.K.L., Craig, A., Boord, P., Kirkup, L., Nguyen, H., 2003. Development of an algorithm for an EEG-based driver fatigue countermeasure. Journal of Safety Research 34 (3), 321–328.
- [100] Lee, J.D., Young, K.L., Regan, M.A. (2008). Defining driver distraction. In: Regan, M.A., Lee, J.D., Young, K.L. (Eds.), Driver Distraction: Theory, Effects, and Mitigation. CRC Press Taylor and Francis Group, Boca Raton, FL, USA, pp. 3140.
- [101] Lee, S. E., Olsen, E. C. B., Wierwille, W. W.: A Comprehensive Examination of Naturalistic Lane-Changes. Report No. DOT HS 809 702. Washington, D.C.: National Highway Traffic Safety Administration, 2004
- [102] Lee, See. Trust in automation: designing for appropriate reliance. 2004.
- [103] Lehman, J. F., Laird, J., Rosenbloom, P.: A gentle introduction to SOAR, an architecture for human cognition. URL <http://ai.eecs.umich.edu/soar/sitemaker/docs/misc/GentleIntroduction-2006.pdf> (last access 17.01.2014). 2006
- [104] Liang, Y. and Lee, J. D., Driver Cognitive Distraction Detection Using Eye Movements. Berlin, Germany: Springer-Verlag, 2008, pp. 285–300.
- [105] Liang, Y., Lee, J. D., and Reyes, M. L., "Nonintrusive detection of driver cognitive distraction in real time using Bayesian networks," Transp. Res. Rec.: J. Transp. Res. Board, vol. 2018, pp. 1–8, 2007.
- [106] Liang, Y., Lee, J. D., Reyes, M. L.: Non-intrusive detection of driver cognitive distraction in real-time using Bayesian networks. Transportation Research Record: Journal of the Transportation Research Board (TRR) 2018, 1–8, 2007
- [107] Liang, Y., Lee, J. D.: A hybrid bayesian network approach to detect driver cognitive distraction. Transportation Research Part C 38, pp. 146-155, 2014
- [108] Liang, Y., Reyes, M. L., and Lee, J. D., 2007, "Real-time detection of driver cognitive distraction using support vector machines," IEEE Trans. Intelligent Transp. Syst., vol. 8, no. 2, pp. 340–350, Jun. 2007.
- [109] Liang, Y., Reyes, M. L., Lee, J. D.: Real-time detection of driver cognitive distraction using Support Vector Machines. IEEE Transactions on Intelligent Transportation Systems 8 (2), 340–350, 2007
- [110] Liang, Y., Reyes, M., and Lee, J., "Real-time detection of driver cognitive distraction using support vector machines," Intelligent Transportation Systems, IEEE Transactions on, vol. 8, no. 2, pp. 340–350, Jun. 2007.
- [111] Liebner, M., Baumann, M., Klanner, F., Stiller, C. (2012). Driver Intent Inference at Urban Intersections using the Intelligent Driver Model. In 2012 Intelligent Vehicles Symposium (IV). Alcalá de Henares, Spain, June 3-7, 2012, IEEE Catalog Number: CFP12IVS-CDR, ISBN: 978-1-4673-2117-4.



## HoliDes

**H**olistic Human Factors **D**esign of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- [112] Liu, A., Pentland, A. (1997). Towards real-time recognition of driver intentions. In Intelligent Transportation System, 1997. ITSC'97, pp. 236-241.
- [113] Lüdtkke, A., Frische, F., & Osterloh, J.-P. (2011). Validation of a Digital Human Model for Predicting Flight Crew- Aircraft Cockpit Interaction. In Tagungsband Berliner Werkstatt für Mensch-Maschine Systeme. Berlin.
- [114] Lüdtkke, A., Osterloh, J.-P., Mioch, T., Rister, F., and Looije, R.: Cognitive modelling of pilot errors and error recovery in flight management tasks. In Proceedings of the 7th Working Conference on Human Error, Safety and Systems Development Systems Development (HESSD), LNCS 5962, pages 54–67. IFIP TC13.5, Springer, 10, 2009
- [115] Lüdtkke, A., Osterloh, J.-P.: „Simulating Perceptive Processes of Pilots to Support System Design“, Human-computer interaction - INTERACT 2009: 12th IFIP TC 13 International Conference. August 24-25, 2009.
- [116] Lüdtkke, A., Weber, L., Osterloh, J.-P., and Wortelen, B.: Modeling pilot and driver behaviour for human error simulation. In HCI International 2009, LNCS 5610–56. Springer, 10, 2009
- [117] Lüdtkke, A.: Kognitive Analyse Formaler Sicherheitskritischer Steuerungssysteme auf Basis eines integrierten Mensch-Maschine Modells, Dissertationen zur Künstlichen Intelligenz (DISKI) Band 288. Akademische Verlagsgesellschaft Aka GmbH, Berlin. ISBN 3-89838-5769, 2004
- [118] Mabuchi, R., Yamade, K. (2011). Study on Driver-Intent Estimation at Yellow Traffic Signal by Using Driving Simulator, pp. 95-100, In 2011 IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, June 5-9, 2011, IEEE Catalog Number: CFP11IVS-CDR, ISBN: 978-1-4577-0889-3.
- [119] Makeig, S., Jung, T., 1995. Changes in alertness are a principal component of variance in the EEG spectrum. Neuroreport 7, 213–216.
- [120] Makeig, S.M., 1993. In low Lapses in alertness: coherence of fluctuations in performance and EEG spectrum. Electroencephalography and Clinical Neurophysiology 86, 23–35.
- [121] Matousek, M.I., Petersen, A., 1983. A method for assessing alertness fluctuations from EEG spectra. Electroencephalography and Clinical Neurophysiology 55, 108–113.
- [122] McCall, J. C., Trivedi, M. (2007). Driver Behavior and Situation Aware Brake Assistance for Intelligent Vehicles. In Proceedings. of the IEEE, vol. 95, no. 2, pp. 374-386.
- [123] McCall, J. C., Trivendi, M. (2006). Human behaviour based predictive brake assistance. In Proceedings of the 2006 IEEE Intelligent Vehicles Symposium, pp. 8-12.
- [124] McCall, J. C., Wipf, D. P., Trivedi, M. M., Rao, B. D. (2007). Lane change intent analysis using robust operators and Sparse Bayesian Learning. In IEEE Transactions on Intelligent Transportation Systems, pp. 431-440.



## HoliDes

**H**olistic Human Factors **D**esign of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- [125] McCracken, J.H., Aldrich, T.B.: Analyses of Selected LHX Mission Functions: Implications for Operator Workload and System Automation Goals. Fort Rucker, AL: U.S. Army Research Institute Aviation Research and Development Activity; Technical Note ASI479-024-84, 1984
- [126] Möbus, C., Eilers, M. (2011). Prototyping Smart Assistance with Bayesian Autonomous Driver Models. In Mastrogiovanni, F. and Chong, N.-Y. (eds), Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspectives, pp. 460-512, IGI Global publications, DOI: 10.4018/978-1-61692-857-5, ISBN13: 978-1-61692-857-5, ISBN10: 1-61692-857-3, EISBN13: 978-1-61692-858-2.
- [127] Morris, B., Doshi, A., Trivedi, M. (2011). Lane Change Intent Prediction for Driver Assistance: On-Road Design and Evaluation. In 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 895-901, Baden-Baden, Germany, June 5-9, 2011, IEEE Catalog Number: CFP11IVS-CDR, ISBN: 978-1-4577-0889-3.
- [128] Myers, B. A. (1990). A New Model for Handling Input. ACM Transactions on Information Systems, Vol. 8, No. 3, pp. 289-320 Proceedings of the Sixth International Conference on Multimodal Interfaces (ICMI '04), State College, PA, USA, October 13 - 15, 2004. ACM Press, New York, NY, pp 212-219.
- [129] Neisser, U.: Cognition and reality: principles and implications of cognitive psychology. W.H.Freeman, San Francisco, 1976
- [130] Newell, A., & Simon, H. A.: GPS, a program that simulates human thought. In Heinz Billing, Hrsg., Lernende Automaten, S. 109-124. Oldenbourg, München, 1961
- [131] Nigay, L. and Coutaz, J. (1993). A design space for multimodal systems: concurrent processing and data fusion. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93)*. ACM, New York, NY, USA, 172-178.
- [132] Ninomija, S.P., Funada, M.F., Yazu, Y., Ide, H., Daimon, N., 1993. Possibility of ECGs to improve reliability of detection system of inclining sleep stages by grouped alpha waves. In: Proceedings of the 15th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 1410-1411.
- [133] Olson, R.L., Hanowski, R.J., Hickman, J.S., and Bocanegra, J. L., "Driver distraction in commercial vehicle operations," Tech. Rep., 2009.
- [134] Ormerod, T. C. and Shepherd, A.: Using task analysis for information requirements specification: The SGT method. Lawrence Erlbaum Associates, 2004
- [135] Ortiz, M. G., Fritsch, J., Kummert, F., Gepperth, A. (2011). Behavior prediction at multiple time-scales in inner-city scenarios. In 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 1066-1071, Baden-Baden, Germany, June 5-9, 2011, IEEE Catalog Number: CFP11IVS-CDR, ISBN: 978-1-4577-0889-3.



## HoliDes

**H**olistic Human Factors **D**esign of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- [136] Osterloh, J.-P., Bracker, H., Müller, H., Kelsch, J., Schneider, B., & Lüdtkke, A.: DCoS-XML: A Modelling Language for Dynamic Distributed Cooperative Systems. In Proceedings of the 2013 11th IEEE International Conference on Industrial Informatics (INDIN), page 774-779, 2013
- [137] Östlund, J., Nilsson, L., Carsten, O., Merat, N., Jamson, H., Jamson, S., Mouta, S., Carvalhais, J., Santos, J., Anttila, V., Sandberg, H., Luoma, J., Waard, D., Brookhuis, K., Johansson, E., Engström, J., Victor, T., Harbluk, J., Janssen, W., and Brouwer, R., "HASTE Deliverable 2 - HMI and Safety-Related Driver Performance, contract no. grd1/2000/25361s12.319626", Aug. 2004.
- [138] Paternò, F.: Model-Based Design and Evaluation of Interactive Applications. Applied Computing. Springer-Verlag, Berlin. ISBN: 185233155, 1999
- [139] Pentland, A. and Liu, A., 1999. "Modeling and prediction of human behavior," *Neural Comput.*, vol. 11, no. 1, pp. 229-242, Jan. 1999.
- [140] Piso, E.: Task analysis for process-control tasks: The method of Annett et al. applied. *Occupational Psychology* 54, 347-254, 1981
- [141] Puterman, M.L.: Markov Decision Processes. Discrete Stochastic Dynamic Programming, Wiley, Chichester, 2005
- [142] Ranney, T., Mazzae, E., Garrott, R. and Goodman, M. (2000). NHTSA driver distraction research: Past, present and future, Nat. Highway Traffic Safety Admin., Washington, DC.
- [143] Rasmussen, J., 1983. Skills, rules and knowledge: signals, signs and symbols and other distinctions in human performance models. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13, 257-266
- [144] Regan, M. A., Young, K. L.: Driver distraction: a review of the literature and recommendations for countermeasure development. In: *Proc. Australas. Road Safety Res. Policing Educ. Conf. 7* (v1), S. 220-227, 2003
- [145] Regan, M. A.; Young, K. L. (2003): Driver distraction: a review of the literature and recommendations for countermeasure development. In: *Proc. Australas. Road Safety Res. Policing Educ. Conf. 7* (v1), S. 220-227.
- [146] Regan, M.A., Hallet, C. and Gordon, C.P. (2011). Driver Distraction and Driver Inattention: definition, relationship and taxonomy. In *Accident Analysis and Prevention journal*, Elsevier, vol. 43.
- [147] Richard, C. M., Campbell, J. L., & Brown, J. L. (2006). Task analysis of intersection driving scenarios: Information processing bottlenecks.
- [148] Roth, B., 1961. The clinical and theoretical importance of EEG rhythms corresponding to states of lowered vigilance. *Electroencephalography and Clinical Neurophysiology* 13, 395-399.
- [149] Salvucci, D. D. (2004). Inferring driver intent: A case study in lane-change detection. In Proceedings of the Human Factors Ergonomics Society 48th Annual meeting, pp. 2228-2231.



## HoliDes

**H**olistic Human Factors **D**esign of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- [150] Salvucci, D. D. (2007). Integrated models of driver behavior. In Gray, W. D. (ed.), *Integrated models of cognitive systems*, pp. 356-367, New York, Oxford University Press.
- [151] Salvucci, D. D., Gray, R. (2004). A two-point visual control model of steering. In *Perception*, 33, pp.1233-1248.
- [152] Sandblom, F., Brännström, M. (2011). Probabilistic Threat Assessment and Driver Modeling in Collision Avoidance Systems, In 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 914-919, Baden-Baden, Germany, June 5-9, 2011, IEEE Catalog Number: CFP11IVS-CDR, ISBN: 978-1-4577-0889-3.
- [153] Sarter, N.B. and Woods, D.: "How in the World Did We Ever Get into That Mode? Mode Error and Awareness in Supervisory Control", *Human Factors - The Journal of the Human Factors and Ergonomics Society*, 37(1):5-19, 1995
- [154] Sebok, A., Wickens, C., Sarter, N., Quesada, S., Socash, C., and Anthon, B.: The automation design advisor tool (adat): Supporting flight deck design in nextgen. In *Proceedings of the Human Factors and Ergonomics Society 54th Annual Meeting*, 2010
- [155] Sottet, J-B. et al. (2007). Model-Driven Adaptation for Plastic User Interfaces. In *Proc. INTERACT 2007, the eleventh IFIP TC13 International Conference on Human-Computer Interaction*, pp. 397-410.
- [156] Stanton, N. A.: Hierarchical task analysis: developments, applications and extensions. *Applied Ergonomics*, 37, (1), 55-79, 2006
- [157] Stein, E.S., 1992. Air Traffic Control Visual Scanning. FAA Technical Report DOT/FAA/CT-TN92/16. US Department of Transportation.
- [158] Su, J., Zhang, H., Ling, Ch., L., Matwin, S. (2008). Discriminative Parameter Learning for Bayesian Networks. In *Proceedings of the 25<sup>th</sup> International Conference on Machine Learning*, Helsinki, Finland.
- [159] Sugeno M. *Industrial applications of fuzzy control*, Elsevier Science Pub. Co., 1985.
- [160] Tango, F., Aras, R. and Pietquin, O., 2010. Batch reinforcement learning for optimizing driving assistance strategies. In *Neural Information Processing Systems (NIPS) 2010 conference*. Vancouver, Canada, December, 10-11.
- [161] Tango, F., Aras, R. and Pietquin, O., 2011. Automation Effects on Driver's Behaviour when integrating a PADAS and a Distraction Classifier. *Human Computer Interaction Conference (HCI2011)*. Orlando, Florida (USA), 09 - 14 July 2011.
- [162] Tango, F., Minin L., Montanari, R., & Botta, M. (2010, August, 16-19). Non-intrusive Detection of Driver Distraction using Machine Learning Algorithms. In the proceeding of the XIX European Conference on Artificial Intelligence (ECAI). Lisbon, Portugal (ISBN: 978-1-60750-605-8; Editor: IOS Press Amsterdam, The Netherlands, The Netherlands).



## HoliDes

Holistic Human Factors Design of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- [163] Treat, J. et al. (1979). Tri-level Study of the Causes of Traffic Accidents: Final Report. volume 1. Technical Report Federal Highway Administration, US DOT.
- [164] Trick, L. M., Enns, J.T.; Mills, J., Vavrik, J.: Paying attention behind the wheel: a framework for studying the role of attention in driving. In: *Theoretical Issues in Ergonomics Science* 5 (5), S. 385–424. DOI: 10.1080/14639220412331298938, 2004
- [165] Trick, Lana M.; Enns, James T.; Mills, Jessica; Vavrik, John (2004): Paying attention behind the wheel: a framework for studying the role of attention in driving. In: *Theoretical Issues in Ergonomics Science* 5 (5), S. 385–424. DOI: 10.1080/14639220412331298938.
- [166] Vorndran, I.. „Unfallentwicklung auf deutschen Straßen 2010“, Statistisches Bundesamt Deutschland - Destatis, 2010
- [167] Vuckovic, A., Radivojevic, V., Chen, A.C., Popovic, D.B., 2002. Automatic recognition of alertness and drowsiness from EEG by artificial neural networks. *Medical Engineering & Physics* 24 (5), 349–360.
- [168] W3C: MBUI - Task Models, W3C Working Group Note 08 April 2014 (<http://www.w3.org/TR/task-models/>), 2014
- [169] Wang, C J.-S., Knippling, R. R. and Goodman, M. J. The role of driver inattention in crashes: New statistics from the 1995 crashworthiness data system, in Proc. 40th AAAM, Vancouver, BC, Canada, 1996, pp. 377–392.
- [170] Weber, L., Steenken, R., Lüdtkke, A.: „Integrated Modeling for Safe Transportation (IMoST2) - Driver Modeling & Simulation“, 4. Berliner Fachtagung Fahrermodellierung. June 13th - 14th. Berlin 2013.
- [171] Weir, D, Chao K. (2007). Review of control theory models for directional and speed control. In Cacciabue, P. C. (ed.), *Modelling Driver Behaviour in Automotive Environments*, pp. 293-311, London, Springer.
- [172] Wickens, Multiple resources and performance prediction, *Theor. Issues in Ergon. Sci.*, Vol. 3, No. 2, 159-177, 2002.
- [173] Wickens, Multiple resources and performance prediction, *Theor. Issues in Ergon. Sci.*, Vol. 3, No. 2, 159-177, 2002.
- [174] Wiest, J., Höffken, M., Kreßel, U., Dietmayer, K. (2012). Probabilistic Trajectory Prediction with Gaussian Mixture Models. In 2012 Intelligent Vehicles Symposium (IV). Alcalá de Henares, Spain, June 3-7, 2012, IEEE Catalog Number: CFP12IVS-CDR, ISBN: 978-1-4673-2117-4.
- [175] Wilson, B.J., Bracewell, T.D., 2000. Alertness monitor using neural networks for EEG analysis. *Proceedings of the 2000 IEEE Signal Processing Society, Workshop on Neural Networks for Signal Processing*, 2, pp. 814–820.
- [176] Wilson, G.F., 1993. Air-to-ground training missions: a psychophysiological workload analysis. *Ergonomics* 36 (9), 1071–1087.
- [177] Wilson, G.F., Fisher, F., 1991. The use of cardiac and eye blink measures to determine flight segments in F4 crews. *Aviation, Space and Environmental Medicine* 62, 959–962.



## HoliDes

**H**olistic Human Factors **D**esign of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

- [178] Woeller, M., Blaschke, C., Schindl, T., Schuller, B., Faerber, B., Mayer, S. and Trefflich, B., 2011. "Online Driver Distraction Detection using long short-term Memory" in IEEE Transaction on Intelligent Transportation System, vol. 12, No. 2, June 2011, IEEE Society Editor.
- [179] Woeller, M., Blaschke, C., Schindl, T., Schuller, B., Faerber, B., Mayer, S. and Trefflich, B. (2011). Online Driver Distraction Detection using long short term Memory in IEEE Transaction on Intelligent Transportation System, vol. 12, No. 2, IEEE Society Editor.
- [180] Wortelen, B., Lüdtkke, A., Baumann, M.: „Integrated Simulation of Attention Distribution and Driving Behavior“, Proceedings of the 22nd Annual Conference on Behavior Representation in Modeling & Simulation. pp 69-76, BRIMS Society 2013
- [181] [www.d3cos-project.eu](http://www.d3cos-project.eu)
- [182] Yanchao Dong, Zhencheng Hu, Member, IEEE, Keiichi Uchimura, and Nobuki Murayama (2011). Driver Inattention Monitoring System for Intelligent Vehicles: A Review, IEEE Transactions on Intelligent Transportation Systems, Vol 12, n2.
- [183] Yangsheng, X., Ka Keung, C. L. (2005). Human Behavior Learning and Transfer, CRC Press Inc.
- [184] Young K. Regan M. and Hammer M. Driver Distraction a review of the Literature. Technical Report No 206, Monash University, accident Research Center, November 2003.
- [185] Zhang, Y., Owechko, Y., and Zhang, J., 2004. Driver cognitive workload estimation: A data-driven perspective. In Proc. IEEE Intell. Transp. Syst. Conf., Washington, DC, 2004, pp. 642–64