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

D2.4 – Modelling Techniques and Tools V1.0

Project Number:	332933
Classification:	Public
Work Package(s):	WP2
Milestone:	M2
Document Version:	V0.1
Issue Date:	13.02.2014
Document Timescale:	Project Start Date: October 1, 2013
Start of the Document:	Month 14
Final version due:	Month 17
Deliverable Overview:	**Main document:** Modelling Techniques and Tools V1.0 (public) **Annex I:** Requirements Update (confidential) **Annex II:** MagicPED Handbook (confidential)
Compiled by:	Osterloh, Jan-Patrick – OFF
Authors:	Bellet, Thierry - IFS Borchers, Svenja - TWT Botta, Marco - UTO Donatelli, Susanna - UTO Eilers, Mark - OFF Feuerstack, Sebastian - OFF Larsen, Morten - AWI Magnaudet, Mathieu - ENA Martin, Denis - TWT Osterloh, Jan-Patrick - OFF Presta, Roberta - SNV Weber, Lars –OFF
Reviewers:	Käthner, David – DLR Sharples, Robert– CAS-UK
Technical Approval:	Jens Gärtner, Airbus Group Innovations
Issue Authorisation:	Sebastian Feuerstack, OFF

DISTRIBUTION LIST		
Copy type[1]	Company and Location	Recipient
T	HoliDes Consortium	all HoliDes Partners
D	ARTMIS JU	Patrick Vandenberghe

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

RECORD OF REVISION		
Date (DD.MM.JJ)	Status Description	Author
08.12.2014	Initial Structure	JPO, OFF
16.01.2015	IFS contribution on COSMODRIVE section	Thierry Bellet, IFS
22.01.2015	Corrections to task modelling chapter	Morten Larsen, AWI
22.01.2015	ENA contribution	Mathieu Magnaudet
23.01.2015	TWT contribution (Section 3.6)	Svenja Borchers, TWT Denis Martin, TWT
23.01.2015	SVN contribution	Roberta Presta, SNV
26.01.2015	TrainingManager, CASCaS, MagicPED	JPO, OFF
27.01.2015	BAD MoB Models	Mark Eilers, OFF
28.01.2015	HEE contribution	Sebastian Feuerstack, OFF
28.01.2015	Review, Reference fix, Integration	JPO, OFF
28.01.2015	UTO contribution	Susanna Donatelli, UTO
30.01.2015	Requirement Analysis Update, Summary	JPO, OFF
11.02.2015	HEE Section update based on review comments received	SF, OFF
11.02.2015	Task modelling section updated based on review comments.	Morten Larsen, AWI
12.02.2015	Update after comments	Mathieu Magnaudet ENA
12.02.2015	Update after comments	SNV

Table of Contents

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. Model-based approaches can be very helpful to manage this 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. it can be checked for consistency, safety (formal methods) or efficiency.

In WP2, we will develop 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.

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 allow for easier testing and thus improvement of the product quality, while gaining shorter development times 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, target of 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.

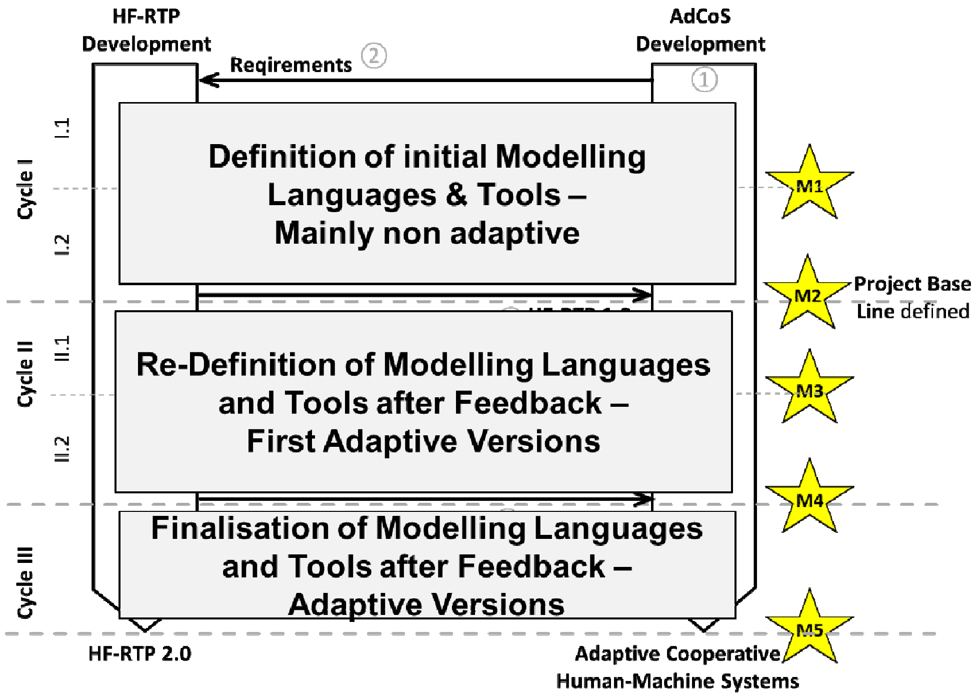

HF-RTP Development
AdCoS Development
Reqirements
2
1
Cycle I
I.1
I.2
Definition of initial Modelling Languages & Tools – Mainly non adaptive
M1
M2
Project Base Line defined
Cycle II
II.1
II.2
Re-Definition of Modelling Languages and Tools after Feedback – First Adaptive Versions
M3
M4
Cycle III
Finalisation of Modelling Languages and Tools after Feedback – Adaptive Versions
M5
HF-RTP 2.0
Adaptive Cooperative Human-Machine Systems

Figure 1: HoliDes Cycled Approach

2 Common Modelling Framework

2.1 Modelling Languages

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 into the common modelling framework.

2.1.1 Task Model

2.1.1.1 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. By a state is meant 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 to the state of the mainly the controlled entity, secondarily the AdCoS, in case of a task aimed at changing the AdCoS in some way, for instance by selecting the information to display 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".

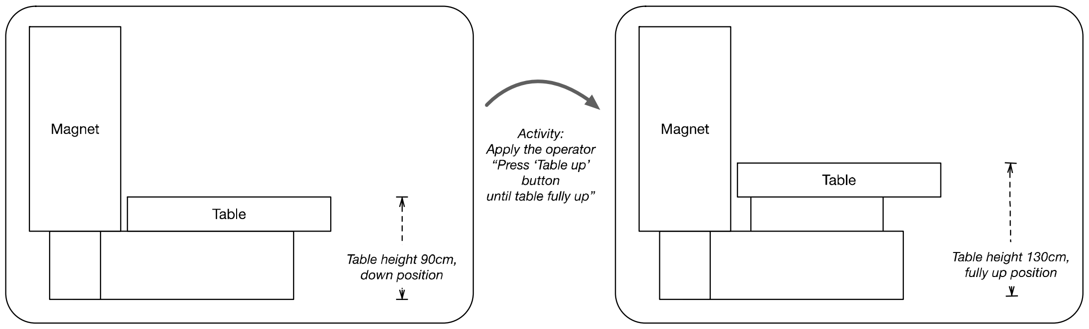

Magnet
Table
Table height 90cm, down position
Activity: Apply the operator "Press 'Table up' button until table fully up"
Magnet
Table
Table height 130cm, fully up position

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.

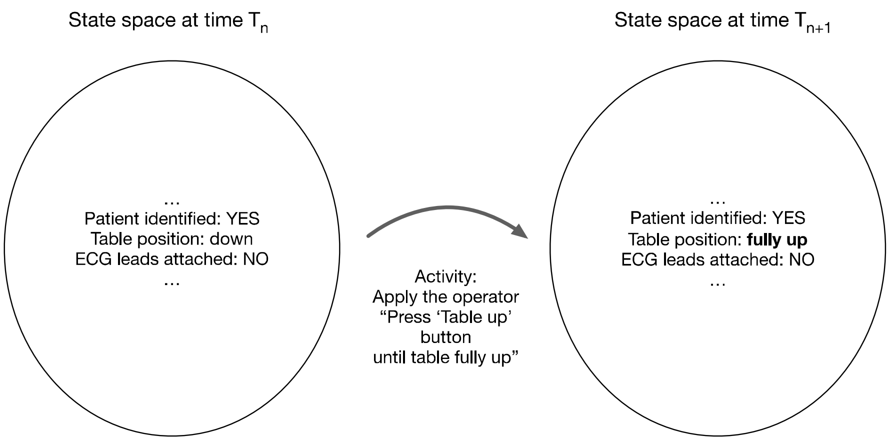


Figure 3: A state space defined by state variables depicted at two moments, before and after a state transition that causes a single variable to change.

As it 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.

2.1.1.2 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 [89] 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 [79], Hodgkinson & Crawshaw [35] and Bruseberg & Shepherd [21], 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 [89] covers the following application areas:

- Training Design
- Interface Design
- Job Design

For each area, the authors have identified the questions listed in the tables below, 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 below, this has been recorded in the Notes column.

Training design		
Knowledge elicitation question	***Category***	***Notes***
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	• State space	

and what feedback is provided?	• 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

Interface Design		
Knowledge elicitation question	***Category***	***Notes***
What are the sensory inputs?	• Information flow	
What information is displayed on the UI	• Information flow	Adapted
What are the information processing demands?	• Information flow • Cognitive load	
What kind of responses are required?	• Information flow • Decision making	
Which control inputs are provided?	• Task operator • Information flow	Adapted
What kind of feedback is given?	• Information flow	
How do the control inputs relate to the goals?	• Goal-means hierarchy	Adapted
Which environmental disturbances are present	• Error handling	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.

Job Design		
Knowledge elicitation question	***Category***	***Notes***
How does information flow in the task?	• Information flow	
When must tasks be done?	• Task planning	
What is the temporal relation of tasks?	• Task sequencing	
What are the physical constraints on tasks?	• Outside constraints	
Where can and cannot error and delay be tolerated?	• Error handling	

What is the cognitive workload	• Cognitive load	Adapted
What are the knowledge requirements for this task?	• Skills-Rules-Knowledge requirements	Adapted
What are the skills requirements for this task?	• Skills-Rules-Knowledge requirements	Adapted

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

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.

2.1.1.3 Application of task models in HoliDes

The following section shows some examples from the AdCoS work packages on how task analysis has been used.

2.1.1.3.1 WP6 - Healthcare

In WP6 task models are used to describe the current workflow of the 3D acquisition use case. Figure 4 depicts one excerpt of the overall 3D acquisition use case that currently contains 148 tasks in total, from which 105 have been identified as basic tasks.

The task analysis is the basis for elaborating a shared understanding between the AdCoS and the MTT. The next step in modelling will be the application of the HEE. For doing this a sequence of basic tasks is currently identified that then will be demonstrated using the HEE's script generation feature to predict the overall task performance of an average user.

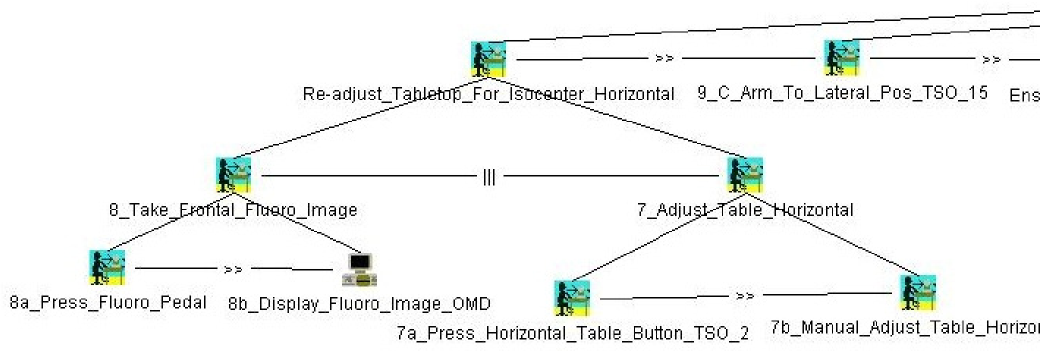

Re-adjust_Tabletop_For_Isocenter_Horizontal
>>
9_C_Arm_To_Lateral_Pos_TSO_15
>>
Ens
8_Take_Frontal_Fluoro_Image
|||
7_Adjust_Table_Horizontal
8a_Press_Fluoro_Pedal
>>
8b_Display_Fluoro_Image_OMD
7a_Press_Horizontal_Table_Button_TSO_2
>>
7b_Manual_Adjust_Table_Horizoı

Figure 4: Excerpt from the task modelling of the 3D acquisition use case of WP6.

2.1.1.3.2 WP7 - Aeronautics

In WP7, the task models are heavily used for modelling the procedures of the crews in A320 and the B737 for the training AdCoS. There, it will be used to compare the procedures of the different aircrafts, in order to improve the transition training from B737 to A320. The concept is shown in Figure 5; the red boxes depict tools from WP2, the blue ones models or tools developed in WP7.

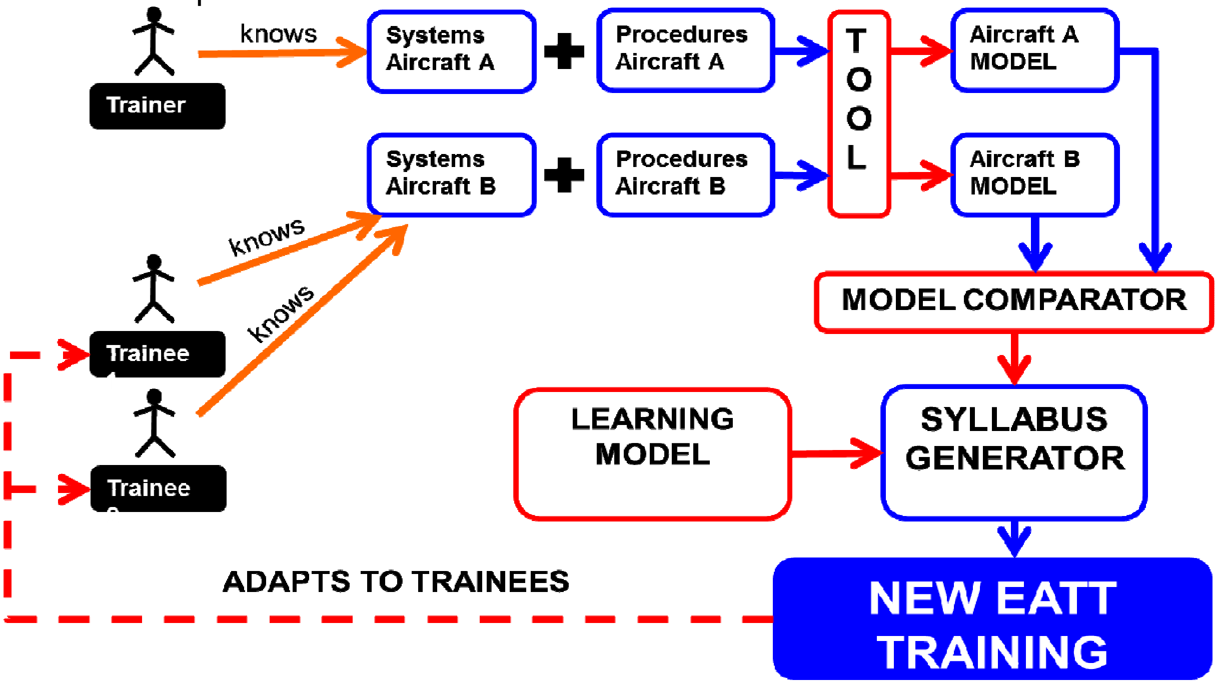


Figure 5: EATT architecture

2.1.1.3.3 WP9 - Automotive

Preliminary task modelling and task analysis have been carried out by REL on Lane Change (LC) Manoeuvre, for the design of the HMI of the mobile app that will implement the LC Assistant.

For the graphical representation of tasks, Microsoft Power Point has been used, i.e. the standard tool used by REL for the task analysis representation. This tool has been selected by REL and it has been included in its HMI development process because it presents several advantages:

- It is cross-domain
- It is flexible (it allows including text, notes, images, schemas, etc..)
- It can be shared with customers (as well as project partners) in order to fasten the definition of the task model

However, it has also relevant drawbacks, such as:

- It does not support the designer in the definition of the task model
- It does not provide any graphical support to create and modify the tasks and subtasks
- The representation of non-trivial tasks (such as the LC Manoeuvre) requires several subtasks and sublevels that could be hardly represented in a single Power Point slide (or even a set of slides).

Therefore, REL decided to apply MagicPED (developed by OFF in WP2) in the next cycle, to assess the potential improvements it can bring. This preliminary modelling activity thus represents the baseline for the future cycles, in order to understand the improvements that could be achieved by using the MagicPED instead of Microsoft Power Point.

The task analysis of lane change manoeuvre provides a general normative description of the tasks involved with making a Lane Change (LC) Manoeuvre. These tasks are preliminary cognitive, motor, visual or some combination thereof. The preliminary task modelling and task analysis carried out by REL LC Manoeuvre was meant as a preparatory activity for the definition of the HMI of the LC Assistant. In fact, the task analysis can highlight the cognitive, visual and motor loads of the driver in each task and subtask of the overall LC Manoeuvre, and this information is key for the design of an HMI that can actually adapt to the context (driving status, driving intention, etc..), and provide tailor-made information which the driver is capable of processing in continuously changing conditions. This task overview has implications for HMI design and prioritization of information according to the specific task the driver is performing, in order to avoid presenting safety critical information or pressing for decision-making tasks where cognitive load is high or, preventively, when lane change is likely to occur (according to the forecast provided by the driver intention module).

The overall LC Manoeuvre task has been split into 3 sub-tasks, as shown in Figure 6.

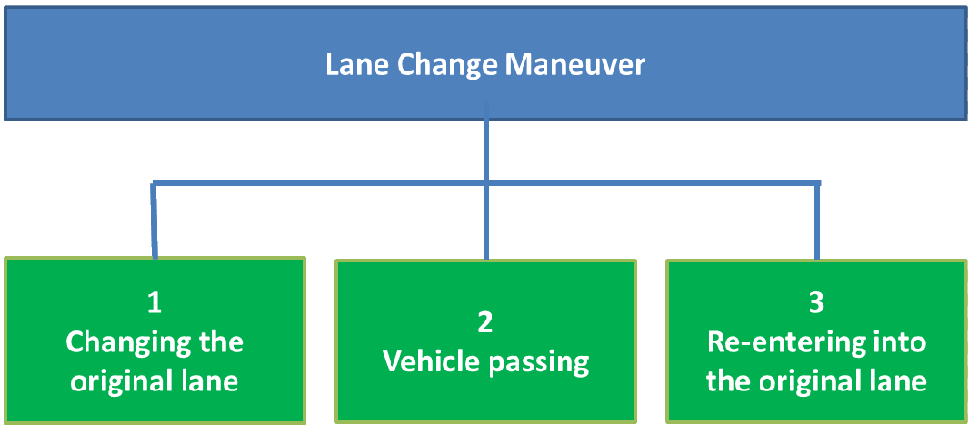

Lane Change Maneuver
1 Changing the original lane
2 Vehicle passing
3 Re-entering into the original lane

Figure 6: Preliminary macro task analysis of Lane Change Manoeuvre

By adapting the guidelines provided by Lee, Olsen and Wierwill [54] for interchange design and sign placement, each sub-subtask can be categorized as:

1) Decision: decide when the manoeuvre is possible (sub-tasks in yellow)
2) Preparation: prepare for the manoeuvre (sub-tasks in orange)
3) Execution: perform the manoeuvre (sub-tasks in blue)

Figure 7 represents the different phases (decision, preparation and execution) while changing the lane.

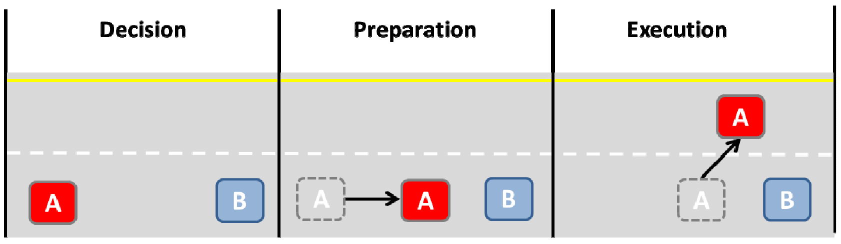


Figure 7: Representation of decision, preparation and execution tasks to change the original lane (task1)

Figure 8 shows the task model for the first subtask (Changing the original lane), where the different colours highlights how it includes decision, preparation and execution tasks.

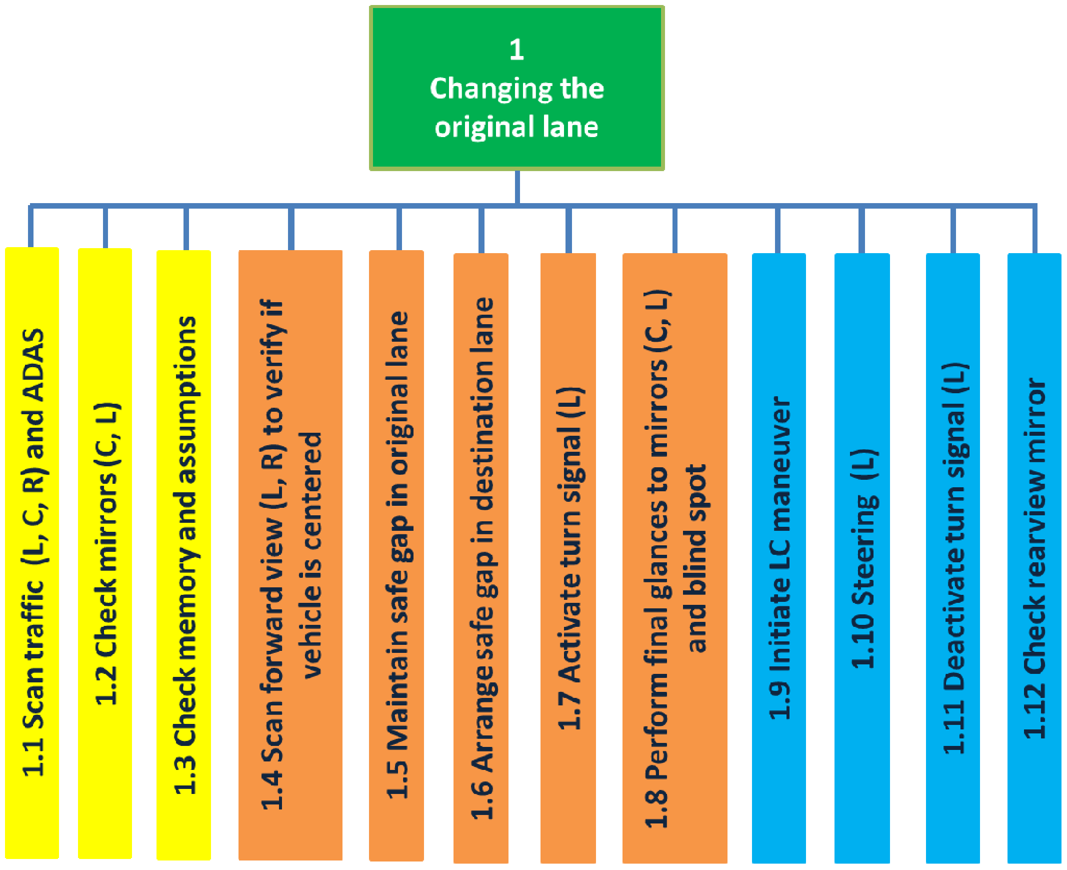

1
Changing the original lane
1.1 Scan traffic (L, C, R) and ADAS
1.2 Check mirrors (C, L)
1.3 Check memory and assumptions
1.4 Scan forward view (L, R) to verify if vehicle is centered
1.5 Maintain safe gap in original lane
1.6 Arrange safe gap in destination lane
1.7 Activate turn signal (L)
1.8 Perform final glances to mirrors (C, L) and blind spot
1.9 Initiate LC maneuver
1.10 Steering (L)
1.11 Deactivate turn signal (L)
1.12 Check rearview mirror

Figure 8: Sub-tasks involved in changing the original lane (task1)

The approach has been applied to the other sub-tasks ("vehicle passing" and "re-entering into the original line").

Figure 9 represents the different phases (decision, preparation and execution) while passing a vehicle (second sub-task), and Figure 10 the corresponding task model.

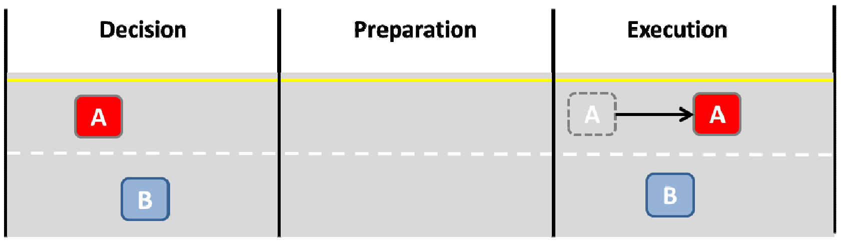


Figure 9: Representation of decision, preparation and execution tasks to pass a vehicle (task2)

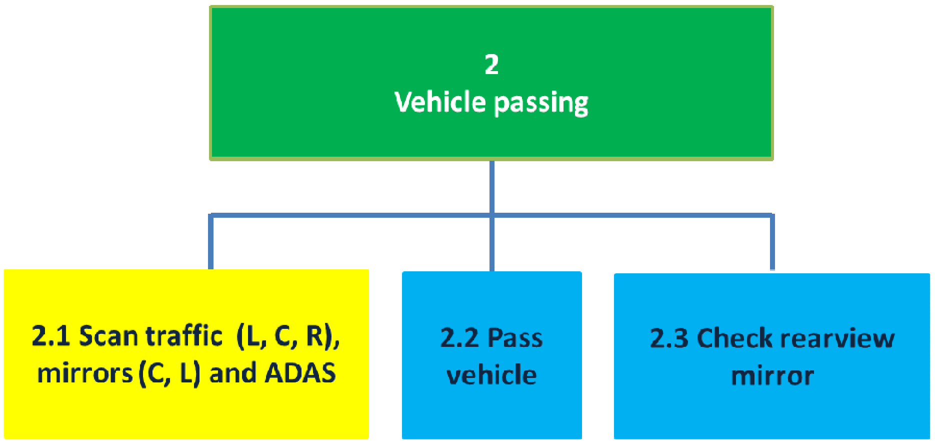

2
Vehicle passing
2.1 Scan traffic (L, C, R), mirrors (C, L) and ADAS
2.2 Pass vehicle
2.3 Check rearview mirror

Figure 10: Sub-tasks involved in passing a vehicle (task2)

Figure 11 represents the different phases (decision, preparation and execution) while re-entering the original lane (third sub-task), and Figure 12 the corresponding task model.

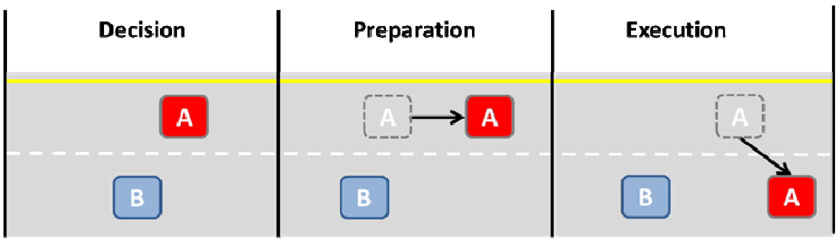


Figure 11: Representation of decision, preparation and execution tasks to re-enter the original lane (task3)

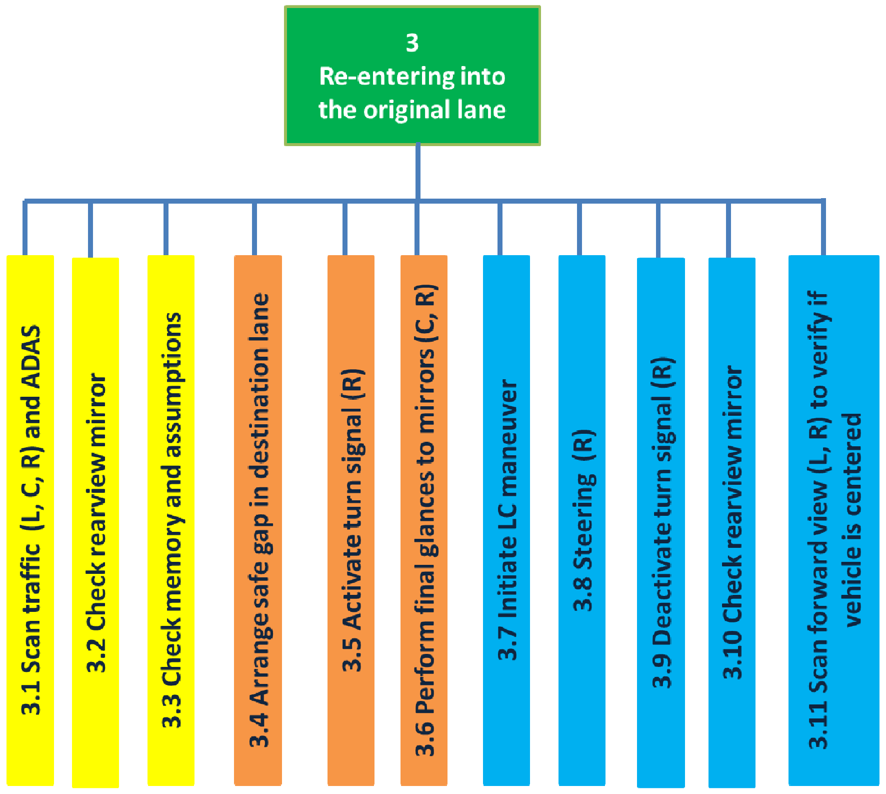

3
Re-entering into the original lane
3.1 Scan traffic (L, C, R) and ADAS
3.2 Check rearview mirror
3.3 Check memory and assumptions
3.4 Arrange safe gap in destination lane
3.5 Activate turn signal (R)
3.6 Perform final glances to mirrors (C, R)
3.7 Initiate LC maneuver
3.8 Steering (R)
3.9 Deactivate turn signal (R)
3.10 Check rearview mirror
3.11 Scan forward view (L, R) to verify if vehicle is centered

Figure 12: Sub-tasks involved in re-entering the original lane (task3)

According to the cognitive, motor and visual tasks that the driver must complete in each phase, he/she has different cognitive, motor and visual loads, summarized in Table 4.

Subtask		Decision	Preparation	Execution
1. Changing the original lane	**Cognitive load**	medium	medium	medium
	Visual load	high	high	medium
2. Vehicle passing	**Cognitive load**	low	low	low
	Visual load	medium	low	medium
3. Re-entering into the	**Cognitive load**	medium	medium	medium
	Visual load	high	high	medium

Table 4: cognitive, motor and visual loads in each subtask.

Table 4 provides a relevant support for the design of the HMI of the AdCoS.

In fact, the AdCoS (LC assistant) can adapt to the status of the driver (distraction, intention, etc.) and the status of the environment (other cars approaching) and provide different information to the driver, by also exploiting different interaction modalities.

The preliminary HMI concept is based on the information included in Table 4. The concept also includes alternative graphics for the adaptive HMI. For more information on the HMI design, refer to deliverable D9.3.

2.1.1.4 Types of task modelling

As can be seen from the knowledge elicitation questions listed previously, 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 [36]) – descriptive, normative and formative models. A brief explanation follows:

2.1.1.4.1 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.

2.1.1.4.2 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 2.1.1.5.1.1 below) and Goals, Operators, Methods, and Selection rules (GOMS, see section 2.1.1.5.1.2 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.

2.1.1.4.3 Formative

The formative approach to task modelling tries to capture what can be done with the system [42]. 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 5.

	Task model		
Type of 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 5: Overview of task model categories. Adapted from [36], p208

2.1.1.5 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:

2.1.1.5.1.1 HTA

HTA – hierarchical task analysis – is widely used in variety of domains and contexts (incl. interface design) [7]. The main characteristics of HTA can be summarized as follows:

1) Work is decomposed into a hierarchical structure:
   - Goals and subgoals – 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
1) 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 [42].

2.1.1.5.1.2 GOMS

The GOMS model was developed by Card, Moran and Newell [23] as a way of quantitatively predicting the skilled and error free performance of users interacting with a text editor, and is now commonly refered 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" ([23], 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.

2.1.1.5.1.3 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 [94], 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 13, 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.

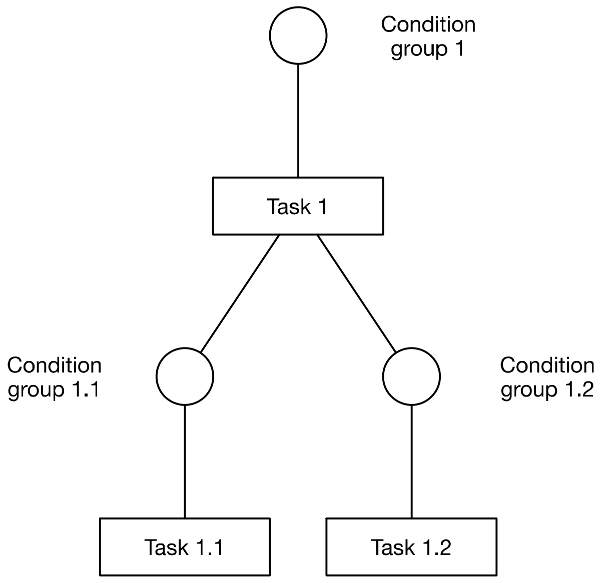


Figure 13: A simple example of tasks and condition groups.

As stated above, the description here is deliberately simple. For details of the meta-model, please refer to [94].

2.1.1.6 HoliDes task modelling language

The HoliDes 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 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 14 shows the task hierarchy as UML Diagram.

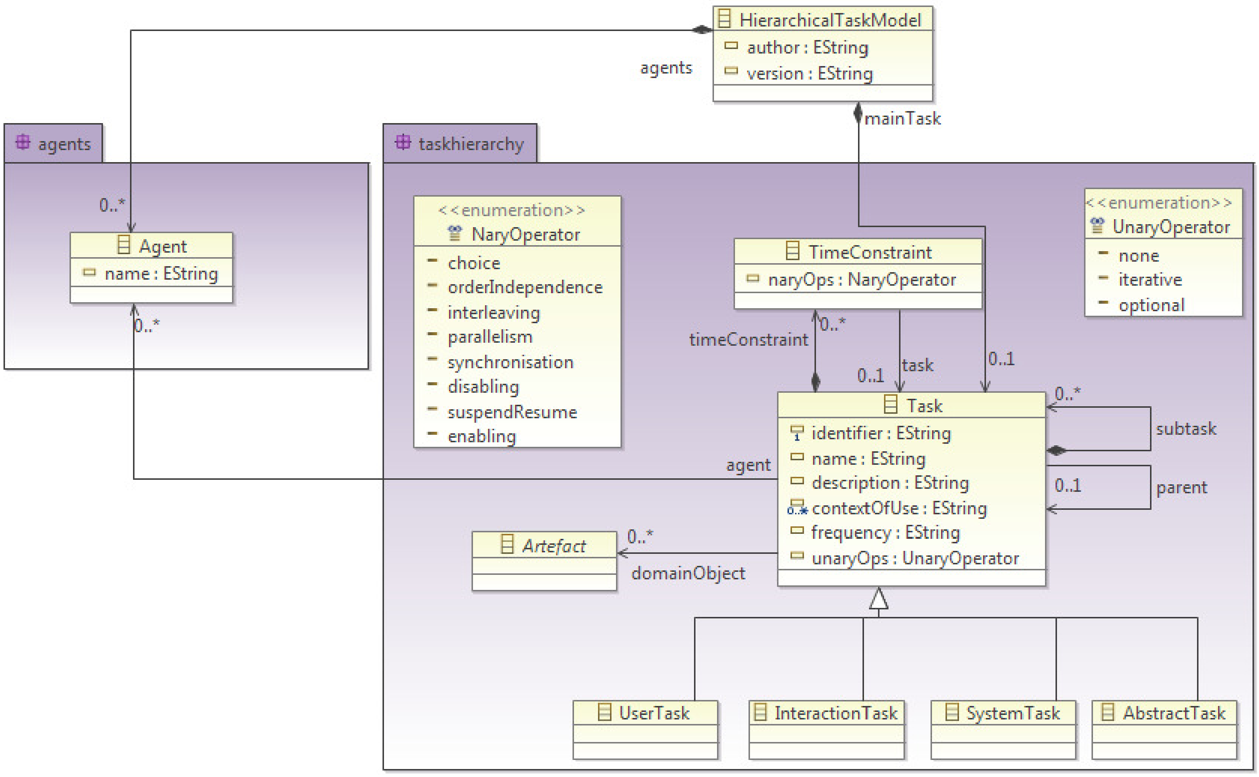


Figure 14: Task Hierarchy Model

Figure 15 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, or b) one of the following analyses provided in future versions should be performed:

- Execution time
- Workload
- Task/Procedure Comparison

Figure 15 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 tasks 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.

2.1.1.6.1 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 which consists of `Retrieve` and `Condition` statements can be matched, the rule is selectable.

2.1.1.6.1.1 Retrieve(variable, age)

The retrieval request searches the memory for the occurrence of the specified *variable*. A variable is a chain of associative links which 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.

2.1.1.6.1.2 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'* ...

2.1.1.6.2 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, which trigger motoric actions move the visual perception or they are used to actively memorize certain values or to add certain relations (associative links) between nodes.

2.1.1.6.2.1 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 an variable, an AOI or an object type matched in the LHS.

2.1.1.6.2.2 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 models 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.

2.1.1.6.2.3 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 value is appended as destination node of the path
3. Is fully matched, a new value is appended to the path.

2.1.1.6.2.4 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.

2.1.1.6.3 Workload Annotation

Each action element in the rules allows adding workload annotations, according to the workload theory of McCracken & Aldrich [70].

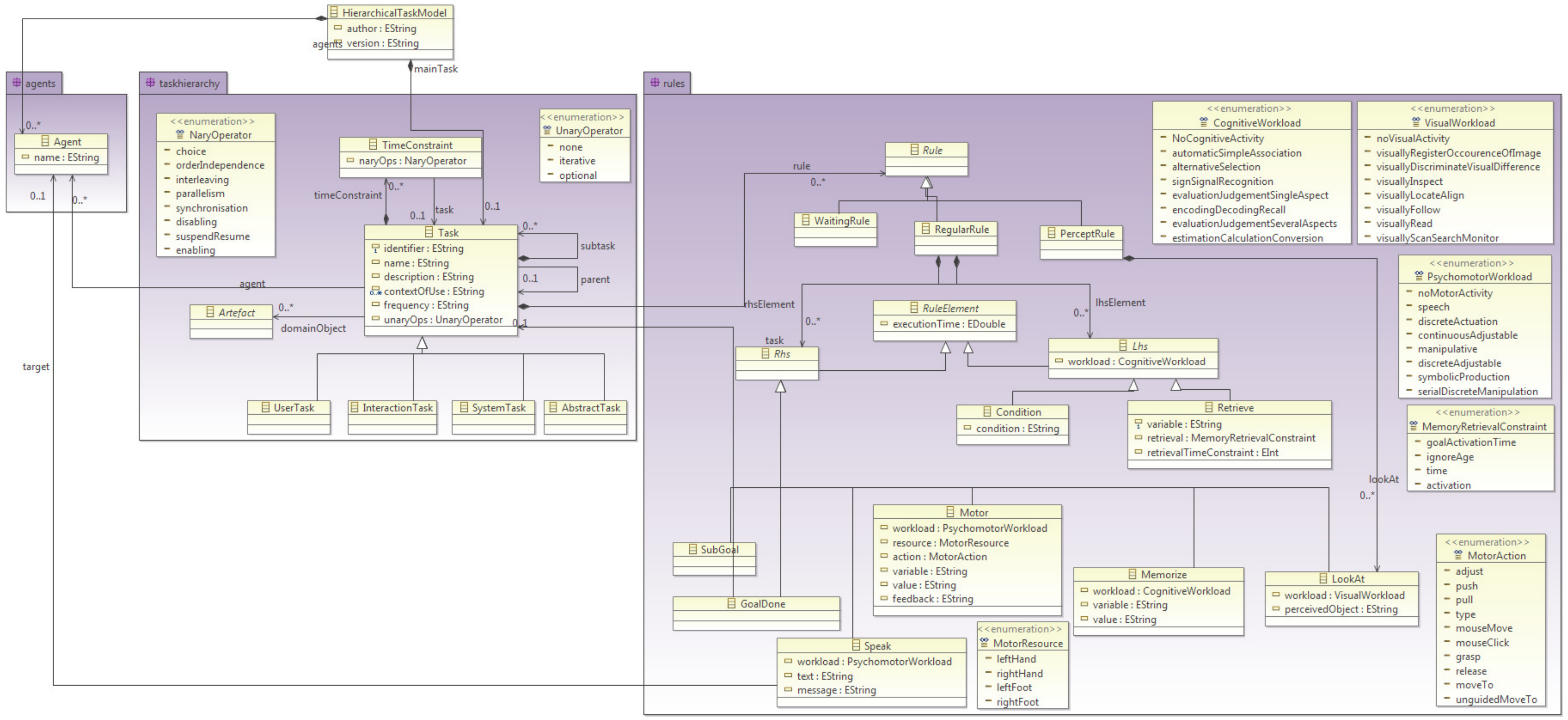


Figure 15: Rule Level Model

2.1.2 Resource Modelling Language

In general, a resource is a source or supply from which benefit is produced, but there are several other definitions, depending on the field a resource is defined in, e.g. in biology a resource are substances or object required by a biological organism for normal maintenance, growth or reproduction. It can also be natural resources (anything from the environment), human resources (skills, energies, talents, knowledge, …), or computer resources (memory capacity, network capacity or speed, CPU availability, …). Due to this wide range for definitions, there is also a wide range of modelling languages and models available for "resources", mostly in the form of one or more mathematical formulas, e.g. for network planning. To our knowledge, there is no modelling language which allows covering all or a even subset of these various definitions/models.

We started to work on the HoliDes Resource Model, which will focus on the "resources" that are needed for the data exchange within the HF-RTP. Figure 16 shows the ecore[2] Model of the 1st, 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 2.1.5.
- The `HardwareArtefact` class represents any resource that is hardware. This has been taken from DCoS-XML. Hardware is currently distinguished as `DiscreteActuator` (e.g. a on-off-button, gear-shift), `ContinuousActuator` (e.g. the altitude selector of an aircraft's autopilot), `Consumable` or `Sensor`s.
- 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-

[2] Eclipse modeling framework (EMF): http://eclipse.org/emf

resources. While the Artefact class and it's 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 an 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.

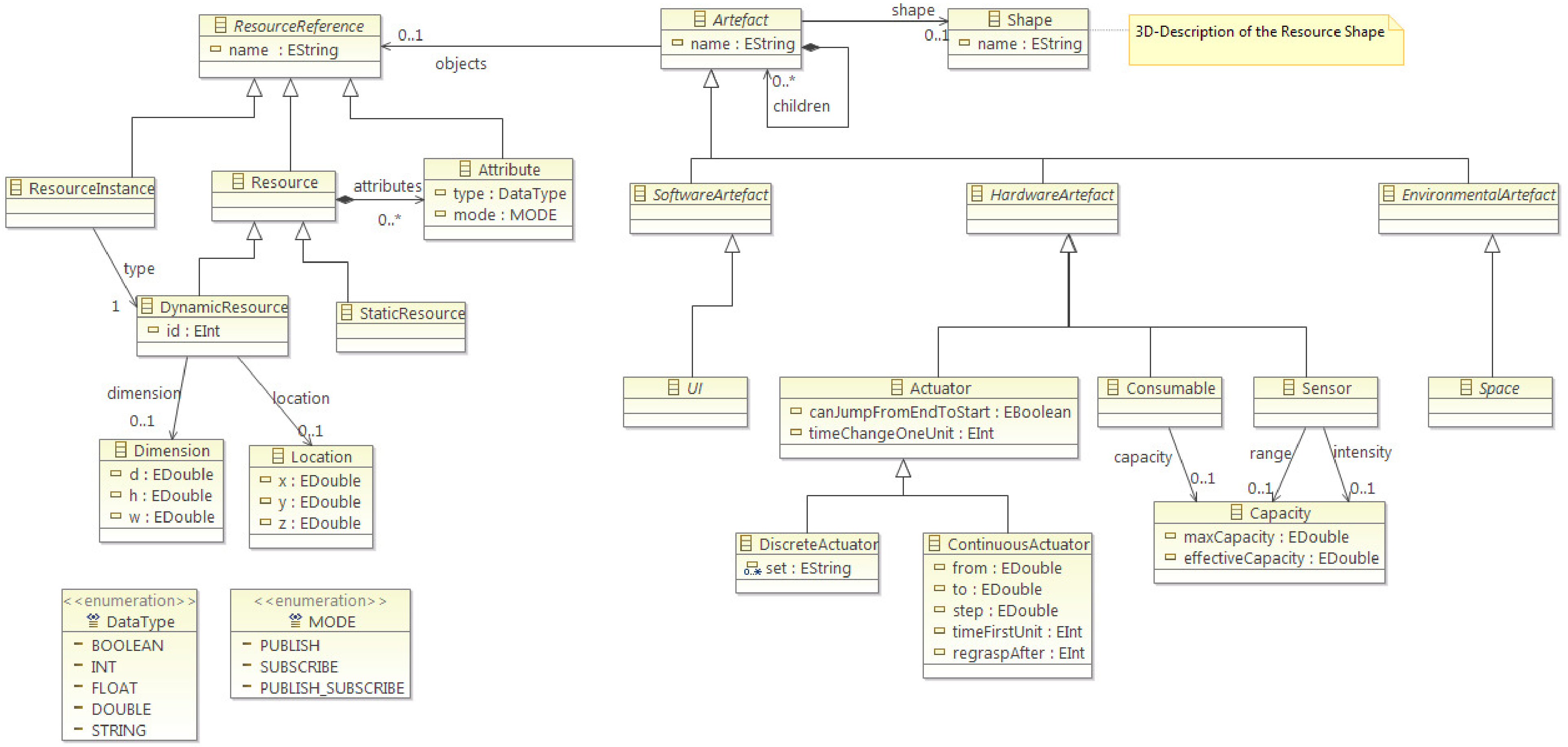


Figure 16: Resource Model

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.

2.1.3 Cooperation Model

Adopting the definition provided in [105], cooperation is an activity of interference management between non-independent tasks distributed among several agents.
Two forms of cooperation are investigated in HoliDes. One relies on the notion of assistance (*assistance paradigm*), when a single human agent is assisted by one or more machine agents, typically to enhance safety and reliability and/or reducing workload. The other involves multiple human and machine agents cooperating in the realization of common super-ordinate tasks, using shared resources (*human-machine system paradigm*). In the *assistance paradigm*, the focus is on a single user, cooperatively assisted by one or more machine agents. In the *human-machine system paradigm*, the focus is on the system of multiple human and machine agents cooperating on the super-ordinate tasks.

The analysis of HoliDes use cases reveals that the two paradigms are typically both involved. For example:

- **health domain**: multiple human agents in the hospital where the patient is treated form a cooperative human-machine system (*human machine system paradigm*), whose super-ordinate task is to diagnose and treat the patient. Many of these agents receive local assistance from machine agents (*assistance paradigm*), for example the MRI examiner with automatic calibration of RF electric field patterns.

- **aeronautic domain**: there are two use cases in the aeronautic domain.
    - airport diversion assistant: two machine agents (on-board system, AOS; MadCoS) and two pilots are in the cockpit and form a cooperative human machine system (*human machine system paradigm*). The two machine agents play an assistive role (*assistance paradigm*): AOS monitors systems and warns in case of malfunctions, MadCoS can take over the task of looking up a diversion airport and present suggestions to the crew. The assistive aspect is therefore prominent in this use case.

- adaptive flight simulator transition training (TRS): the objective of the TRS tool is to assist the instructor in evaluating experience and skills gaps between the trainee's current state and the one expected on the aircraft he or she is transitioning. That gap is monitored all along the training program. This use case therefore exclusively resort from the *assistance paradigm*.

- **control room domain**: a control room is a cooperative human-machine system (*human-machine system paradigm*). A series of operators (human agents) and equipment (machine agents) work together to perform the overall functions of the room (e.g. border surveillance). Within the control room, the relationship between the operators and the machine agents can be assistive (*assistance paradigm*), with for example enhanced vision and detection capabilities, automated surveillance, automated risk evaluation, support to decision-making.

Automotive domain represents the exception since, within the HoliDes project, it is mainly considered from the assistance paradigm perspective, where the drivers in the vehicles receive significant assistance from machine agents (*assistance paradigm*).

HoliDes is focused on the modelling of the collaboration dynamics within AdCos systems.
The concept of collaboration within HoliDes is considered between:

- humans and humans
- humans and machines
- machines and machines

The system is seen as a multi-agent system, where human and machine agents interact with each while pursuing purposes.
The techniques and tools that will be identified/developed will be used for modelling these types of collaboration in the four application domains of the projects, i.e., Healthcare, Aeronautics, Control Rooms and Automotive.

HoliDes collaborative model can leverage as a starting point the research effort about collaboration in multi-agent systems made by the D3CoS Artemis project (Designing Dynamic Distributed Co-operative Human-Machine Systems), since the *adaptive* cooperative systems considered in HoliDes can be seen as a specialization of the distributed cooperative systems addressed by D3CoS. Indeed, the definition given in [77] about Multi-Agent Systems and Cooperation should be imported as-is in the

HoliDes context. As shown in the following, the DCoS-XML modelling language has been thoroughly reviewed in order to evaluate its pros and cons for its adoption and/or extension to address the HoliDes needs.

2.1.3.1 DCoS-XML modelling language

DCoS-XML is a modelling language for the description of distributed cooperative systems that has been developed within the Artemis D3CoS project.
The project focuses on the transportation domain and in particular on the interaction between the human agents and the advanced assistance systems, with the final aim of analysing and reducing the human errors and accidents.
Despite the project's application domains, the DCoS-XML language has been designed to be **domain-independent**.

The reference system, like in the HoliDes case, is a **multi-agent system made by cooperating humans and machines**.
The general reference architecture, indeed, is the same of the HoliDes one, as the main actors can be identified into:

- **agents**: humans or machines;
- **tasks**: goals that are assigned to the agents;
- **resources**: entities that can be used by the agents to complete the tasks;
- **environment**: the boundary conditions where the system evolves within.

The final aim of the DCoS-XML language is describing the static and dynamic properties of such a system in order to study it by model-based simulation.
That goal comprises the modelling of the human agents, tasks and resources.

The implementation choices about the language have been motivated in [77]. Though graphical modelling languages are more intuitive, they lack adequate tools for the automatic processing and a direct textual representation.
Examples of such languages are UML dialects, i.e., UML extensions designed for complex and agent-based systems like SysUML and Agent UML, or other graphical languages like AML.
On the other hand, textual languages are more formal and suitable for automatic processing and for semi-formal documentation and specifications as well.

The most outstanding one is XML and its extensions.
Lots of tools already exist to parse and extract information from descriptive models written in XML. Very often such tools also allow for the direct translation from the XML model to the UML or other graphical representation.
From a tool perspective, the step from XML to UML is much easier than the dual one: indeed, despite of the existence of the XML Metadata Interchange format that can be used for the serialization the UML objects, tool vendors tend to implement it differently, by causing that way interoperability issues.
XML Schema is a flexible and **extensible** language that encapsulates the **object-oriented** paradigm for the definition of data structures.
Focusing on the distributed cooperative systems domain, there are very few XML extensions in literature, and no one successful. For such reason, the modelling language has been developed from scratch.

The main idea behind the DCoS-XML language is the one of providing **a general structure for describing in a semiformal way the distributed cooperative system**. Such a structure can be specialized and extended to better fit the application domain requirements and specifications.
According to such view, the top level elements of the structure have been identified and defined within the DCoS-XML Schema: they are the main actors of a general distributed multi-agent system (i.e., agents, divided into **human agents** and **machine agents**, **environment**, **resources**, **tasks**) and a new entity represented by the **link** model, which allows for the representation of the interconnection characteristics among agents, tasks and resources and of the interconnection properties as well.

A possible tool to be leveraged to work with such a modelling language is **CoSimECS** [77], which is a graphical editor allowing for the composition of a DCoS-XML model by using a GUI instead a plain text editor.
One of the most interesting features of such a tool is the possibility of set up a simulation of the developed model. By means of the tool, it is possible to specify the simulation scenario and to map the agents and resource to concrete simulators, by this way creating the set up for a distributed simulation. For example, human agents can be simulated with CASCAS and resources with **Matlab Simulink**. The distributed simulation is implemented by leveraging the **IEEE HLA** standard interfaces.

By looking at the XML Schema definitions, it seems that the top level elements are polarized to be used within a transport system, i.e., some built-in features of the general models are **transport-specific**.

For example, the general **Agent model** has properties like *acceleration, velocity, position, vehicle,* and *perspective,* which expresses if the agent is *on-board* or in the *traffic.*
The Agent model is specialized into the **Human Agent model** and into the **Machine Agent model**.
While the Machine Agent model does not exhibit specific features besides the basic Agent model, the Human Agent has properties like the *cognitiveLoad* and the *situationAwareness*.
The **Task model** represents tasks that can be hierarchically structured, i.e., they should be decomposable into more connected sub-tasks and a mapping to the task model leveraged by the Magic Draw Procedure Editor (MagicPED) tool is smoothly achievable.
The **Resource model** presents lots of parameters in order to be able to represent anything useful to serve to the task for both human and machine agents. It can represent consumable resources (like fuel, for example) that can be exploited by machine agents, or actuators controlled by human agents.
The **Environment model** is very general as well. The features of such model are static variables, like the description of the physical space where the systems can operate within, and dynamic variables, like for example weather conditions.
The **Link model** allows specifying the connection and the connection properties that link agents, tasks and resources.

2.1.3.2 Conclusion about the adoption of the DCoS-XML modelling language

To the aim of surveying the state of the art about the modelling of **cooperation in multi-agent systems**, the **DCoS-XML language** has been considered. The language is designed for the description of the main actors involved in the system (human agents, machine agents, tasks, resources and environment, as well as their interconnections).

Being XML-based, the modelling language is suitable for **automatic processing, object-oriented, flexible and extensible**. Such properties fit well with the HoliDes requirements.

DCoS-XML has been conceived to be a cross-domain language to be applied in the transportation area, i.e., suitable for different scenarios in that field: Manned Aircraft, Unmanned Aerial Vehicles, Automotive and Maritime. These fields are indeed the application areas of the D3CoS project, where DCoS XML has been developed. Because of that reason, some features of the model are transport-specific and do not fit well with

all the HoliDes application domains, except for the Automotive and Aeronautics ones.
The HoliDes modelling language will then be a generalized version of the DCoS-XML able to make the general models (Agents, Resources, Tasks, Link, and Environment) actually domain-independent for the different HoliDes contexts. Such a general language will be exploited to derive specialized models able to capture the domain-specific properties according on the application needs, as a natural exploitation of the XML-language extensibility. As a proof of concept, the specialized models will be used to represent the cooperation in the selected use cases of the project.

2.1.4 Human Operator Models

As described in the description of work, in HoliDes we differentiate between two types of human operator models. First the cognitive models and second the human behaviour models. Human behaviour models are used for the development of AdCoS, while the cognitive models are more focused on the evaluation of an AdCoS in a later development state. In HoliDes two different cognitive models will be used, CASCaS and COSMODRIVE. These tools will be described in more detail in section 3. How these tools can fit into the Human Operator Models and the Modelling languages especially is currently under discussion in WP2. As the cognitive models will be used in later stages, we will focus on the Human behaviour models in this deliverable.

As depicted in the description of work, *human behaviour models* are intended to model the *overt* behaviour of human operators in cooperative systems. This is in contrast to *cognitive models, which* are intended to model the *covert* mental or cognitive processes of human operators. However, this description is somehow misleading, as the behavioural models discussed in this section do include some covert or hidden aspects of human behaviour and are primarily used to obtain information about these hidden aspects. For example, in HoliDes, we will use behavioural models of the human operator based on *Dynamic Bayesian Networks* (DBNs) that can provide an AdCoS application with information about the hidden intentions of a human operator during runtime.

2.1.4.1 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) [80], a well-known formalism defined by Bellman in the early sixties for studying optimization problems.

2.1.4.1.1 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) [11] 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 $<Comp^{pr}, Comp^{nd}, N^{pr}, N^{nd}>$ 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 $<P,T^{pr},I^{pr},O^{pr},H^{pr},prio^{pr},m_0>$, 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: $T^{pr} \rightarrow 2^{Comppr}$ 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 $<P,T^{nd},I^{nd},O^{nd},H^{nd},prio^{nd},m_0>$ 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.

2.1.4.1.2 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.

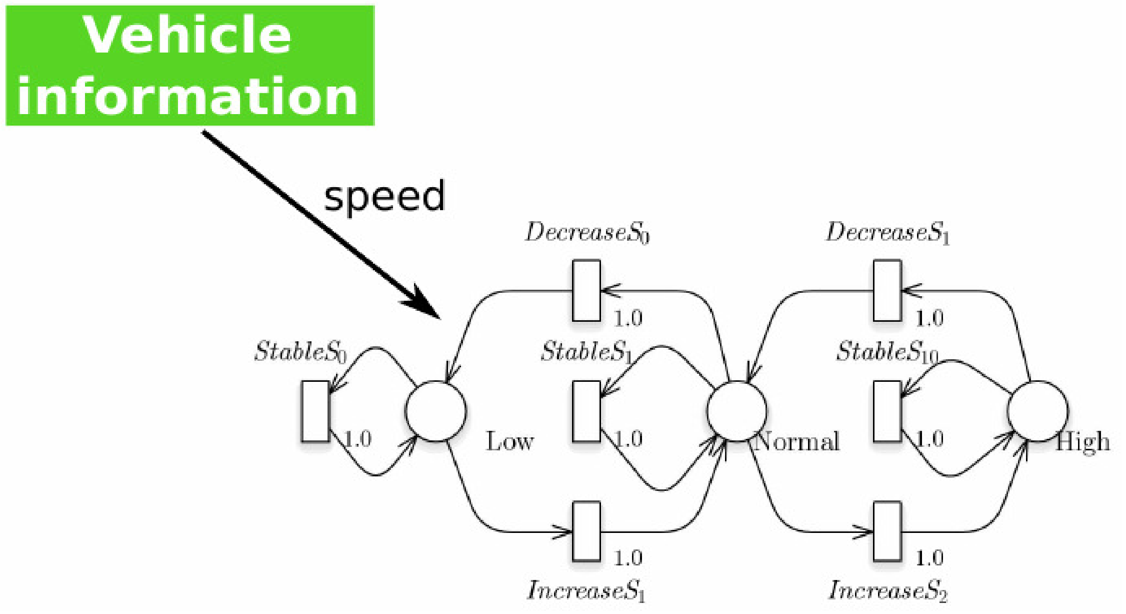


Figure 17: MDPN model for vehicle component

Figure 17 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.

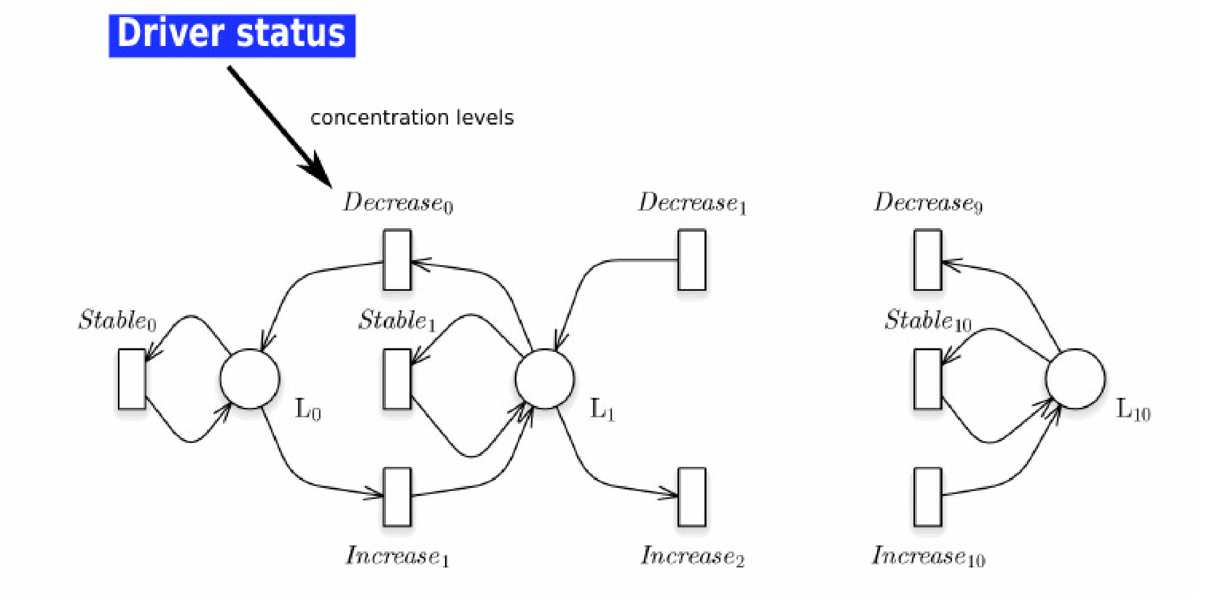


Figure 18: MDPN model for driver component

Figure 18 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,...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 19, 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$_i$.*

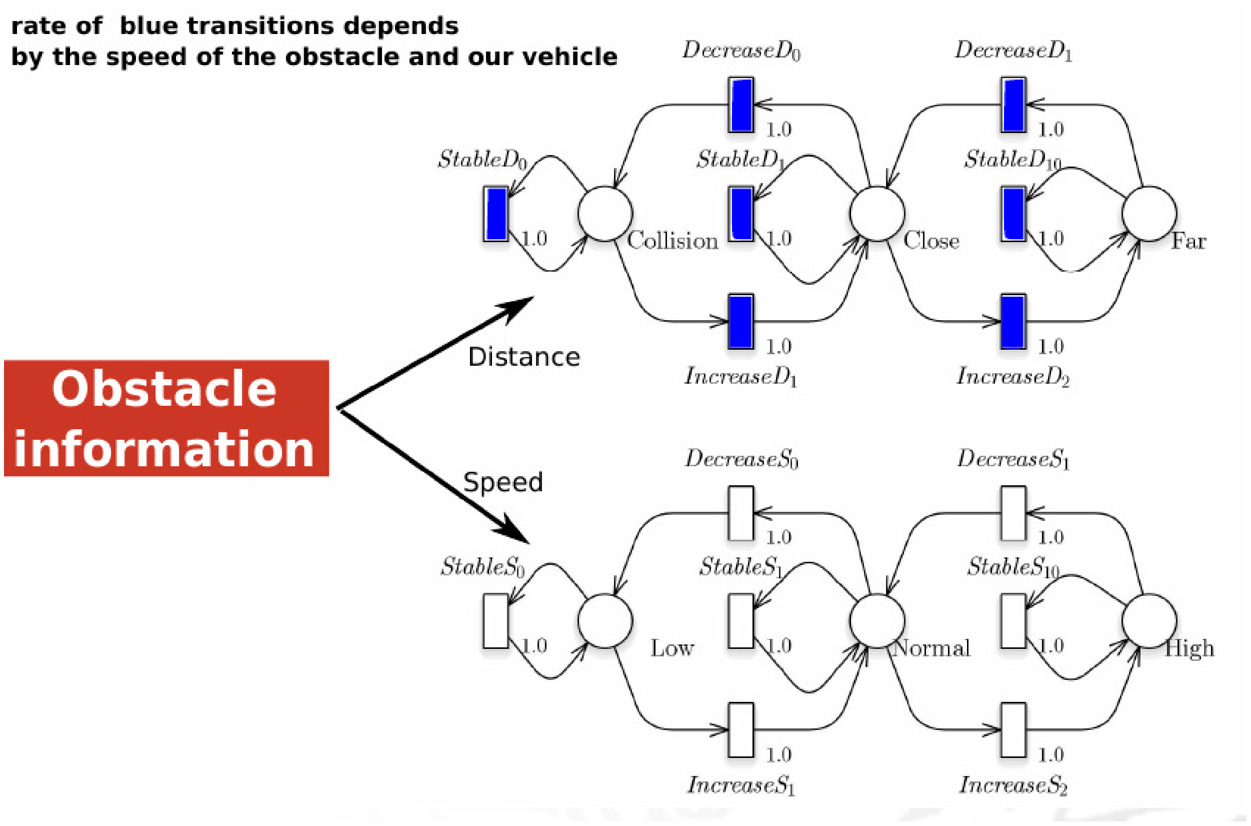
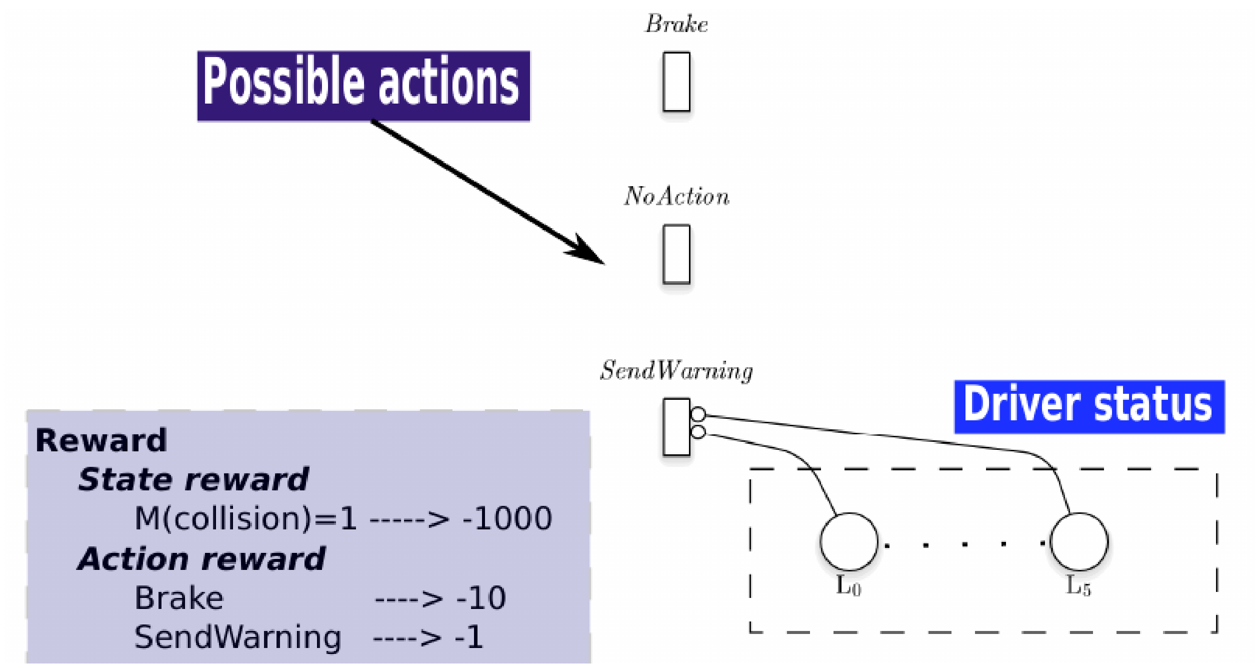


Figure 19: MDPN model for a single obstacle component



Reward
State reward
M(collision)=1 -----> -1000
Action reward
Brake ----> -10
SendWarning ----> -1

Figure 20: MDPN model for action component

Figure 20 depicts an example of **action component**, the nondeterministic Transitions *Brake, 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 *CostCollision* ≫*CostBreak* ≥ *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.

2.1.4.2 The DBN driver model

2.1.4.2.1 (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 $Val(X)$. We use boldface type capital letters $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ to denote sets of random variables (e.g., $\mathbf{X} = \{X_1, \dots, X_n\}$) and corresponding boldface lowercase letters $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ to denote assignments of values to the variables in these sets (e.g., $\mathbf{x} = \{x_1, \dots, x_n\}$). 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_i^t to represent the instantiation of a variable X_i at time t. A sequence $X_i^j, X_i^{j+1}, \dots, X_i^k$ will be denoted by $X_i^{j:k}$ and we will use the notation $x_i^{j:k}$ 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 $x - \epsilon \leq X \leq x + \varepsilon$. When ε is sufficiently small, the probability $P(x - \epsilon \leq X \leq x + \varepsilon)$ can be approximated by

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

Using the same ε for all probabilistic density functions will result in a common pre-factor 2ε 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 PFDs.

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

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

DBNs extend BNs to model the stochastic evolution over a set of variables $\boldsymbol{X} = \{X_1, \dots, X_n\}$ 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 $Val(X_i)$. A DBN D is defined as a pair $D = \{B^1, B^{\rightarrow}\}$, where $B^1 = \{G^1, \theta^1\}$ is a BN that defines the probability distribution $P(\boldsymbol{X}^1)$ and, under the assumption of first-order

Markov and stationary processes, $B^{\rightarrow} = \{G^{\rightarrow}, \theta^{\rightarrow}\}$ is a two-slice Bayesian network (2TBN) that defines the CPD $P(\mathcal{X}^t | \mathcal{X}^{t-1})$ 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 $P\left(X_i^t | \boldsymbol{Pa}(X_i^t)\right)$, where a parent $X_j^t \in \boldsymbol{Pa}(X_i^t)$ can either be in time-slice t or $t-1$. The JPD $P(\mathcal{X}^{1:T})$ over an arbitrary number of T time-slices is then given by the factorization:

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

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 $\boldsymbol{E} \subseteq \mathcal{X}$ of random variables in the model, and an instantiation $\boldsymbol{e}$ to these variables, called the *evidence*, and a subset $\boldsymbol{Y} \subseteq \mathcal{X}$ of variables in the model called the query variables, with $\boldsymbol{Y} \cap \boldsymbol{E} = \emptyset$. Inference then denotes the computation of the posterior probability distribution over the values $\boldsymbol{y}$ of $\boldsymbol{Y}$, conditioned on the fact that $\boldsymbol{E} = \boldsymbol{e}$: $P(\boldsymbol{Y} | \boldsymbol{E} = \boldsymbol{e})$.

Oftentimes, the set of query variables $\boldsymbol{Y} \subseteq \mathcal{X}$ and evidence variables $\boldsymbol{E} \subseteq \mathcal{X}$ are already fixed during design time, i.e., the model will be used to only answer a limited set of fixed queries $P(\boldsymbol{Y} | \boldsymbol{E} = \boldsymbol{e})$. Especially if the potential factorization $P(\boldsymbol{E} | \boldsymbol{Pa}(\boldsymbol{E}))$ 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 $P(\mathcal{X} \setminus \boldsymbol{E} | \boldsymbol{e})$. 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 $P(\mathcal{X} \setminus \boldsymbol{E} | \boldsymbol{e})$ 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.

2.1.4.2.2 DBNs as Human Behaviour Models

Although, described in more detail in Section 3.7, 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 $\boldsymbol{A} = \{A_1, \dots, A_n\}$ representing his different *actions* that compose said behaviours, and a set of discrete and continuous variables $\boldsymbol{O} = \{O_1, \dots, O_m\}$ 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 $P(I^{1:T}, B^{1:T}, \boldsymbol{A}^{1:T}, \boldsymbol{O}^{1:T})$ according to a factorization

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

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

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

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: $P(I^t, B^t | \boldsymbol{a}^{0:t}, \boldsymbol{o}^{0:t})$. Given this joint belief state, we can easily obtain the marginal belief states of intentions $P(I^t | \boldsymbol{a}^{0:t}, \boldsymbol{o}^{0:t})$ and behaviours $P(B^t | \boldsymbol{a}^{0:t}, \boldsymbol{o}^{0:t})$. The estimation of the belief state is known as filtering and can be solved by in constant time by recursively computing $P(I^t, B^t | \boldsymbol{a}^{0:t}, \boldsymbol{o}^{0:t})$ from the past belief state $P(I^{t-1}, B^{t-1} | \boldsymbol{a}^{0:t-1}, \boldsymbol{o}^{0:t-1})$.

2.1.4.2.3 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 3.7.3.2.1). 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 $\boldsymbol{\mathcal{X}} = \{X_1, \dots, X_n\}$.
2. Under the assumption of a first-order Markovian system (3.7.3.2.1), a factorization for the initial time-slice $P(\boldsymbol{\mathcal{X}}^1|\boldsymbol{Y}^1)$, where $\boldsymbol{Y}^1$ may denote the empty set of parents
$$P(\boldsymbol{\mathcal{X}}^1|\boldsymbol{Y}^1) = \prod_{i=1}^{n} P\left(X_i^1|\boldsymbol{Pa}(X_t^1)\right),$$
and a factorization for the 2TBN $P(\boldsymbol{\mathcal{X}}^t|\boldsymbol{\mathcal{X}}^{t-1}, \boldsymbol{Y}^t)$, where $\boldsymbol{Y}^t$ may denote the empty set of parents
$$P(\boldsymbol{\mathcal{X}}^t|\boldsymbol{\mathcal{X}}^{t-1}, \boldsymbol{Y}^t) = \prod_{i=1}^{n} P\left(X_i^t|\boldsymbol{Pa}(X_t^1)\right).$$
3. A set of parameters $\theta_{X,\boldsymbol{Pa}(X)}$ for each factor $P\left(X|\boldsymbol{Pa}(X)\right)$ that is sufficient to identify a function $f\left(x, \boldsymbol{pa}(x), \theta_{X,\boldsymbol{Pa}(X)}\right) = P\left(x|\boldsymbol{pa}(x)\right)$ that given $\boldsymbol{pa}(x)$ and x, returns the conditional probability $P\left(x|\boldsymbol{pa}(x)\right)$ according to the parameters $\theta_{X,\boldsymbol{Pa}(X)}$.

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 $f\left(x, \boldsymbol{pa}(x), \theta_{X,\boldsymbol{Pa}(X)}\right)$ 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 $f\left(x, \boldsymbol{pa}(x), \theta_{X,\boldsymbol{Pa}(X)}\right)$ we need for human behaviour models in HoliDes. As these models consists 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 2.1.4.2.3.1. 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 2.1.4.2.3.5 that allows to reformulate such dependencies. For the fourth case, we restrict our modelling language to Gaussians, described in Section 2.1.4.2.3.2, and mixture of Gaussians, described in Section 2.1.4.2.3.3. For the last two cases, we restrict our modelling language to conditional linear Gaussians, described in Section 2.1.4.2.3.4.

2.1.4.2.3.1 Conditional probability tables (Table-CPDs)

Let X be a discrete variable, and the set of parents $\boldsymbol{Pa}(X)$ be composed of only discrete variables (including the empty set of parents), we can represent any CPD $P(X|\boldsymbol{Pa}(X))$ by a table of probabilities $P\left(x|\boldsymbol{pa}(X)\right)$ for each combination of $x \in X$ and $\boldsymbol{pa}(X) \in \boldsymbol{Pa}(X)$. Such a table can easily be specified by a set of parameters $\theta_{X,\boldsymbol{Pa}(X)}$ with an entry $\theta_{x,\boldsymbol{pa}(x)} = P\left(x|\boldsymbol{pa}(X)\right)$ for each combination of $x \in X$ and $\boldsymbol{pa}(X) \in \boldsymbol{Pa}(X)$.

2.1.4.2.3.2 (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 [51]. The Gaussian PDF for a continuous variable X is fully specified by a mean μ and a variance σ^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 $\boldsymbol{U} = \{U_1, \dots, U_m\}$, we need to specify a set of parameters $\theta_{X,\boldsymbol{Pa}(X)}$ with different means and variances for each combination of parent values $\boldsymbol{pa}(X) \in \boldsymbol{Pa}(X)$:

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

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 $\boldsymbol{X} = \{X_1,\dots,X_n\}$ is commonly characterized by an n-dimensional mean vector $\boldsymbol{\mu} = (\mu_1,\dots,\mu_n)$ and a symmetric $n\times n$ covariance matrix Σ. Let Σ^{-1} denote the inverse covariance matrix and $|\Sigma|$ denote the determinant of Σ, the pdf is defined as:

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

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.

2.1.4.2.3.3 (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 [51]. A mixture of Gaussians over a single continuous variable X, composed of a finite number of n Gaussians, defines a density

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

where $\boldsymbol{\Psi} = (\mu_1,\dots\mu_n,\sigma_1^2,\dots,\sigma_n^2,\lambda_1,..\lambda_{n-1})$ 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

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

where $\boldsymbol{\Psi} = (\boldsymbol{\mu}_1,\dots\boldsymbol{\mu}_n,\Sigma_1,\dots,\Sigma_n,\lambda_1,..\lambda_{n-1})$ 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.

2.1.4.2.3.4 Conditional linear Gaussians

In the case of a continuous variable X with continuous parents $\boldsymbol{Pa}(X)$, we restrict the resulting PDF to *linear Gaussian CPDs* [51]. Let $\boldsymbol{Pa}(X) = \{Y_1, \dots, Y_n\}$, a linear Gaussian CPD is defined by a set of parameters $\beta_0, \dots, \beta_n$ and a variance σ^2 with the PDF:

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

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

2.1.4.2.3.5 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 [51] provides an alternative approach to represent a CPD $P(X|\boldsymbol{Y}, \boldsymbol{Z})$:

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

This allows us to approximate the complex dependencies stated in $P(X|\boldsymbol{Y}, \boldsymbol{Z})$ by an alternative simpler factorization of $P(X, \boldsymbol{Y}|\boldsymbol{Z})$ (e.g., $P(X, \boldsymbol{Y}|\boldsymbol{Z}) = P(X|\boldsymbol{Z})P(\boldsymbol{Y}|X, \boldsymbol{Z})$), 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 $P(X, \boldsymbol{Y}|\boldsymbol{Z})$ for different values of $\boldsymbol{Z}$, which allows to encode context-specific independencies [51].

2.1.5 HMI Interaction Models

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 SCADE 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 either to the weakness of their expressive power or 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. We argue that 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.

Since several years, ENA is 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. 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:

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

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 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 be 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. It also makes it easier to structure applications as collections of reusable components.

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 21 below 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 21: 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.

2.1.6 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 22 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 `Requirement`s, 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 2.1.1) 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 3.4).

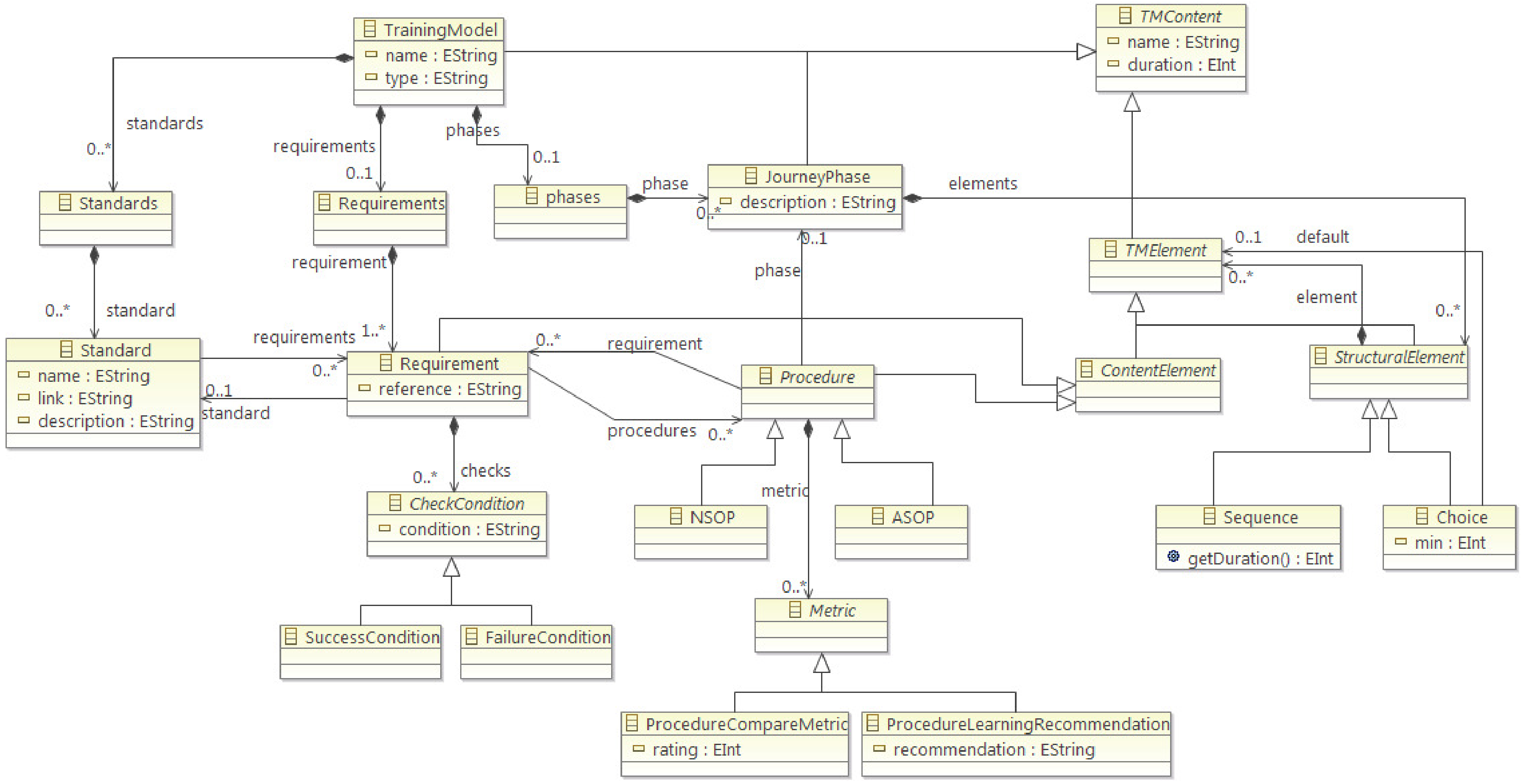


Figure 22: Training Model

2.2 Methods, Techniques and Tools Overview

				WP6					WP7		WP8		WP9					
				Safe patient transfer	Guided Patient positioning	Safe parallel transmit scanning	iXR 3D Acquisition	Patient data access	Diversion Assistance	Transition Training	Border Control Room	Energy Control Room	Frontal Collision Scenario	Overtaking				
Tool	**Extended Name / Description**	**Used Model**	**Partner**	**PHI**	**PHI**	**UMC**	**PHI**	**ATO**	**HON**	**TRS**	**CAS**	**IRN**	**IAS**	**IAS**	**DLR**	**IFS**	**TAK**	**CRF**
CASCaS	Cognitive Architecture for Safety Critical Task Simulation	Cognitive Model	OFF									(X)	(X)				X	
COSMODRIVE	Cognitive Simulation Model of the car Driver	Cognitive Model	IFS										(X)			X		
djnn	HMI model editor	UI / Interaction Model	ENA				X		X			(X)						
BAD-MoB	Bayesian Autonomous Driver Mixture-of-Behaviors Models, Driver state inference / behaviour prediction	Human Behaviour Model	OFF															X
Driver distraction model	Model of human (audio) distraction	Human Behaviour Model	TWT										X	(X)	X		X	X
GreatSPN for MDPN	Editor for MDPN used as virtual co-pilot	Petri Net Model	UTO		X	X			X			X	X					
HEE	Human Efficiency Evaluator	Task Model, Cognitive Model (CASCaS)	OFF				X				X							
MagicPED	Procedure Editor extension of MagicDraw UML Tool	Task Model	OFF							X								
Training Manager	Tool for creation of adapted training syllabi	Task Model, Training Model	OFF							X								

3 Modelling Techniques and Tools V1.0

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 rational 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 2.1.1.1.

In previous projects, OFFIS 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, OFFIS 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 OFFIS wanted to concentrate more on conceptual work and algorithm development, instead on editor development, it has been decided to base future development on COTS UML tools. COTS UML tools can be extended via UML profiles, and often also via plugins. For OFFIS it seems more practicable to concentrate on development of the UML profile and plugins, because UML is widely used in the industry, and 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 Unified Modelling Language (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, OFFIS has chosen MagicDraw (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 [34] the CTT notation abstracts from system-related aspects to avoid a representation of implementation details. Paternò [78] 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 and 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. Further on, 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 2.1.1.5.1.3.

The editor for CTT is called CTTE (ConcurTaskTrees Environment). Figure 23 shows a screenshot of the editor.

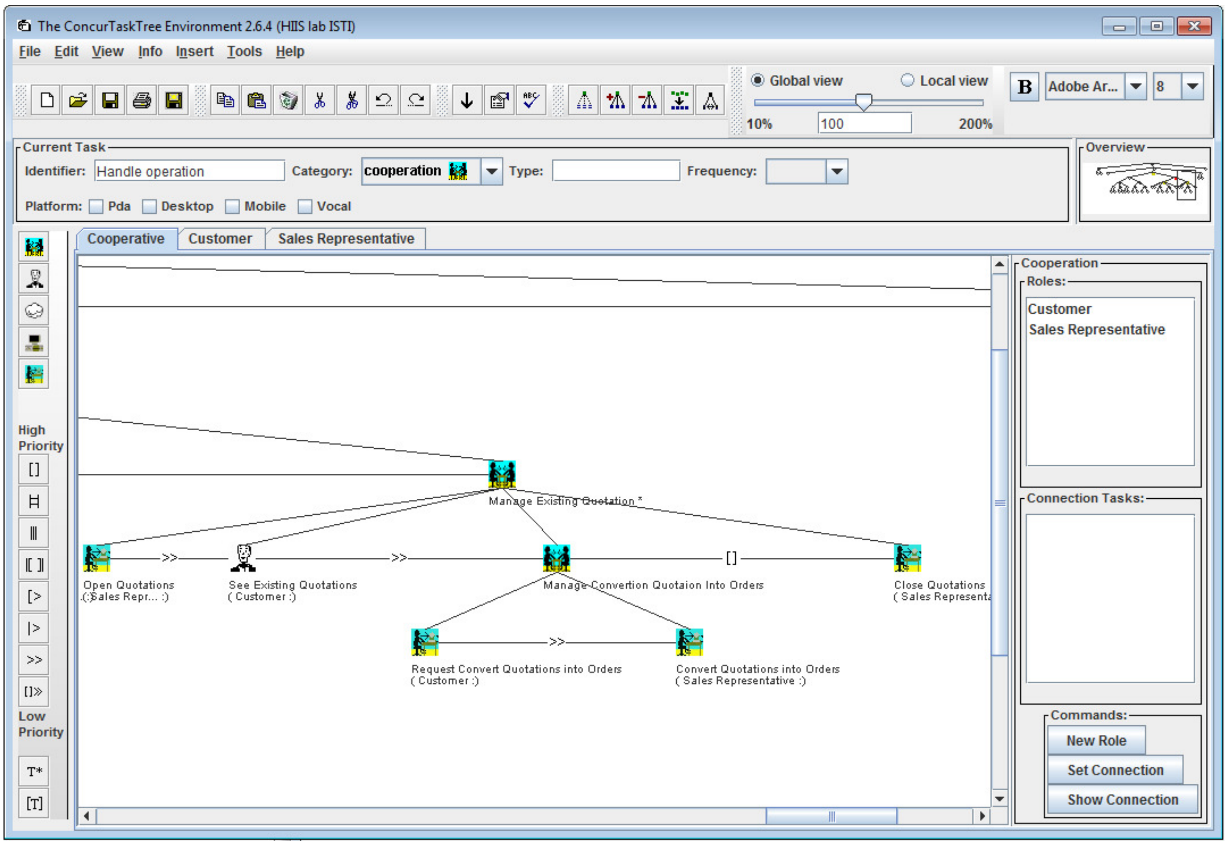

The ConcurTaskTree Environment 2.6.4 (HIIS lab ISTI)
File Edit View Info Insert Tools Help
Global view
Local view
10%
100
200%
Adobe Ar...
8
Current Task
Identifier: Handle operation
Category: cooperation
Type:
Frequency:
Platform: Pda Desktop Mobile Vocal
Overview
Cooperative
Customer
Sales Representative
High Priority
Low Priority
Manage Existing Quotation *
Open Quotations (Sales Repr... :)
See Existing Quotations (Customer :)
Manage Convertion Quotaion Into Orders
Close Quotations (Sales Representa
Request Convert Quotations into Orders (Customer :)
Convert Quotations into Orders (Sales Representative :)
Cooperation
Roles:
Customer
Sales Representative
Connection Tasks:
Commands:
New Role
Set Connection
Show Connection

Figure 23: 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 [23] 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 its derivates 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[3], 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 OFFIS decided to base its task editor on a commercial UML tool: MagicDraw by NoMagic Inc[4]. Therefore, MagicPED consists of two parts. First the UML editor MagicDraw itself, and second a package by OFFIS 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 OFFIS 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 2.1.1.6.
- Rule Diagram: This diagram refers to the GSM based part of the HoliDes task model, see section 2.1.1.6.

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:

[3] Cogulator: http://cogulator.io/
[4] http://www.nomagic.com/products/magicdraw.html

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
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 24 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 25 shows the editor component with an opened Task Hierarchy Diagram, and Figure 26 shows and example for a Rule Diagram.

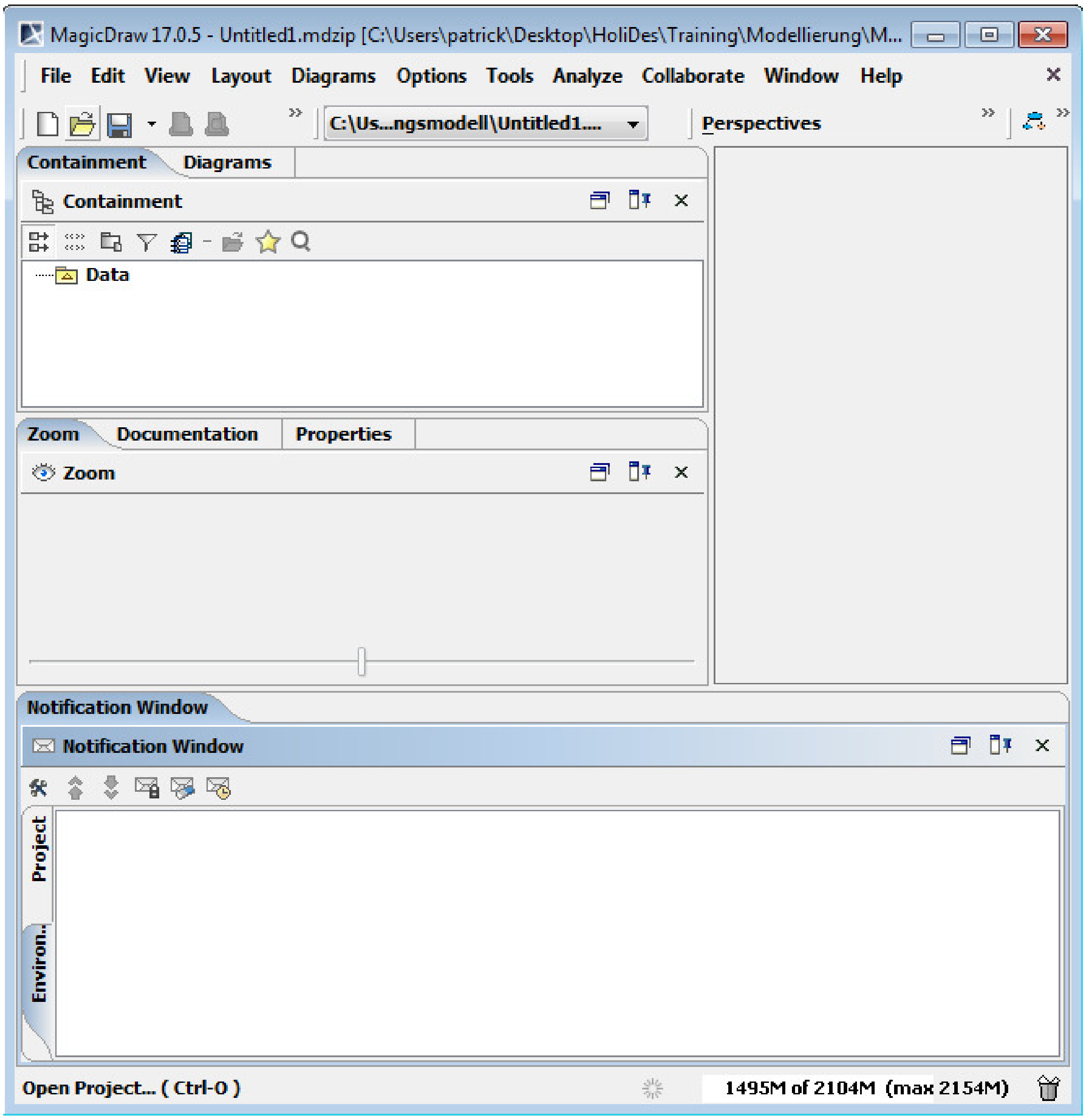

MagicDraw 17.0.5 - Untitled1.mdzip [C:\Users\patrick\Desktop\HoliDes\Training\Modellierung\M...
File Edit View Layout Diagrams Options Tools Analyze Collaborate Window Help
C:\Us...ngsmodell\Untitled1....
Perspectives
Containment
Diagrams
Containment
Data
Zoom
Documentation
Properties
Zoom
Notification Window
Notification Window
Project
Environ...
Open Project... (Ctrl-O)
1495M of 2104M (max 2154M)

Figure 24: MagicDraw Main Window

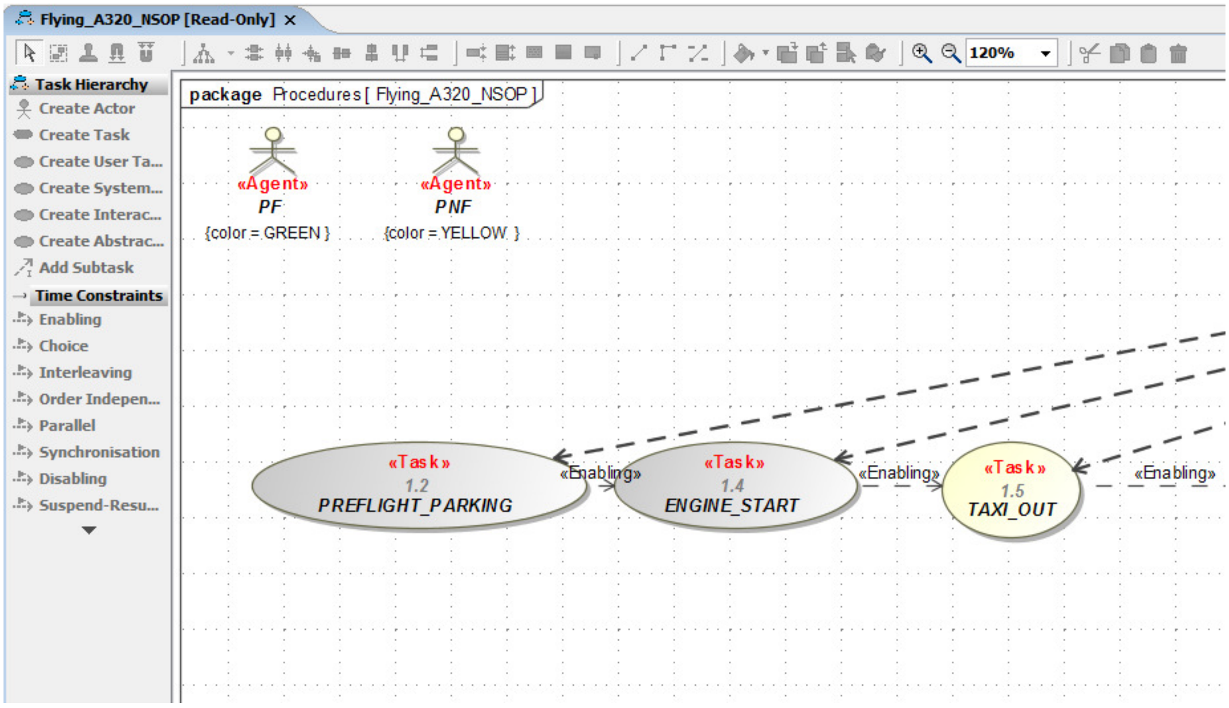


Figure 25: Editor part with Task Hierarchy Diagram

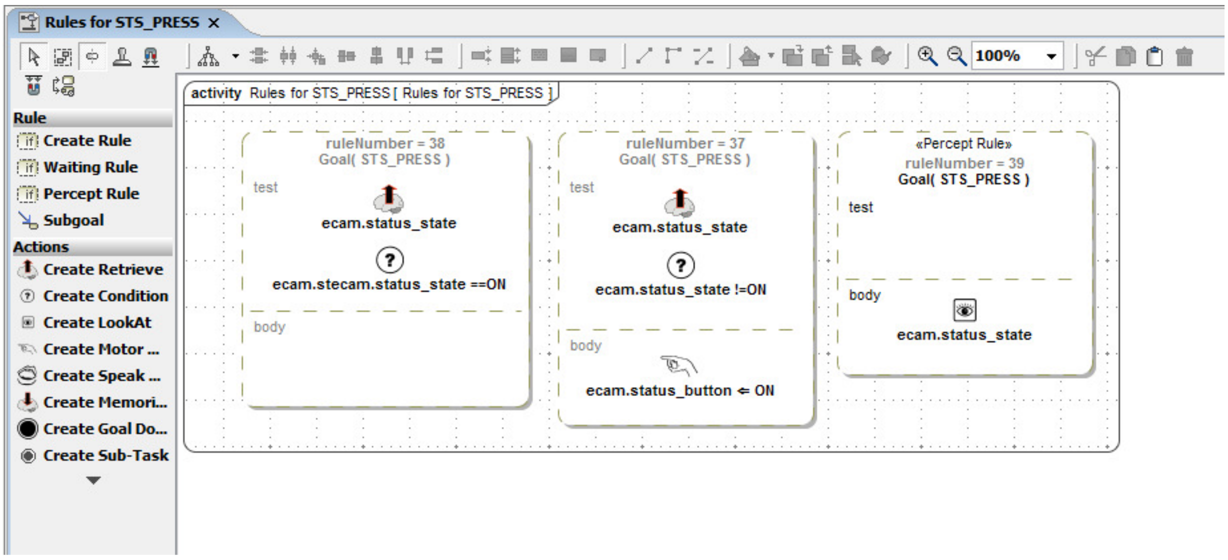


Figure 26: Rule Diagram

More details can be found in the MagicPED manual.

3.1.4 RTP Integration Plan

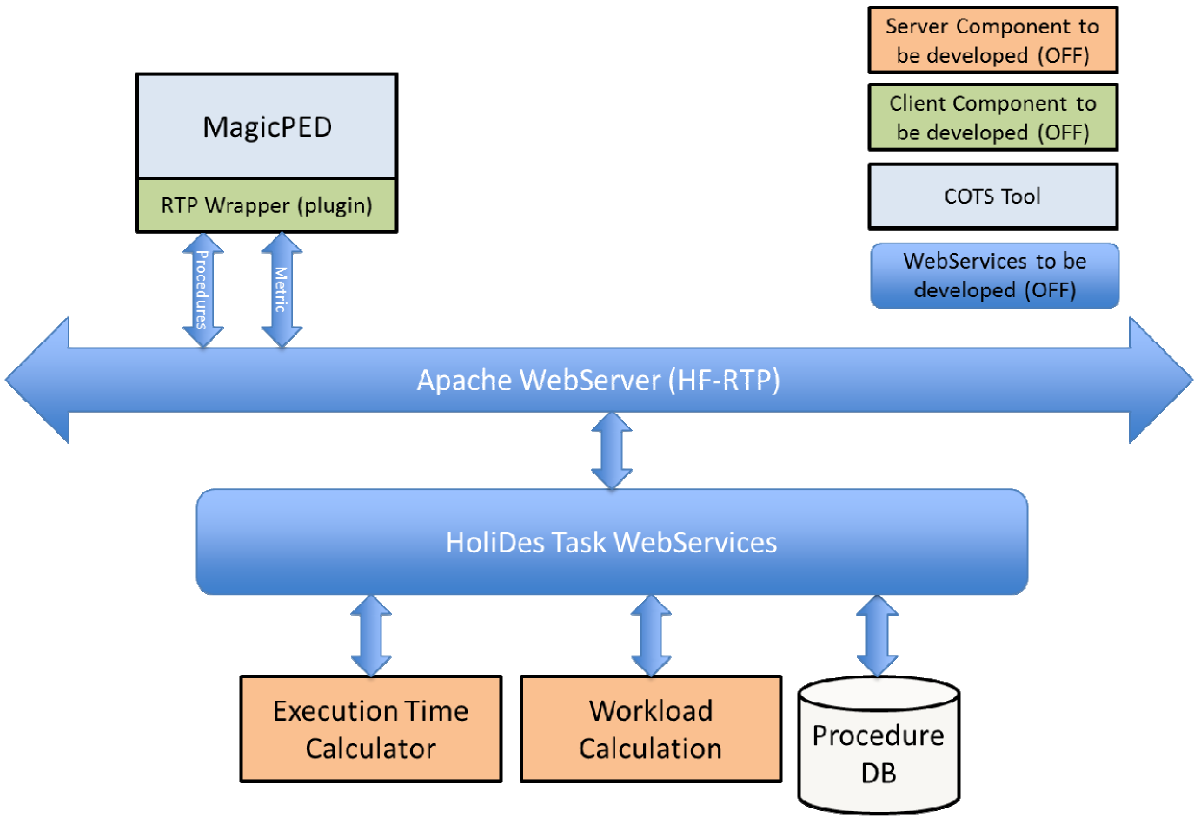

Server Component to be developed (OFF)
Client Component to be developed (OFF)
COTS Tool
WebServices to be developed (OFF)
MagicPED
RTP Wrapper (plugin)
Procedures
Metric
Apache WebServer (HF-RTP)
HoliDes Task WebServices
Execution Time Calculator
Workload Calculation
Procedure DB

Figure 27: MagicPED HF-RTP integration

As described above, in future versions some algorithms for analysing the task hierarchy are planned, e.g. a GOMS like calculation of execution times, or a workload computation. In order to provide these algorithms also to other COTS UML Tools, it is planned to provide these algorithms as part of the HF-RTP. For this, MagicPED will be extended with a plugin, which uses the HF-RTP web-services provided by OFFIS. The plugin will be implemented by OFFIS using the eclipse LYO library[5], which is a library providing OSLC for Java tools. On the other hand, OFFIS will implement an OSLC compliant web-service, which allows to

- Store and retrieve procedures from a DB
- Calculate the execution time of a procedure

[5] http://www.eclipse.org/lyo/

- Calculate the workload of a procedure

There will be further services for the Training Manager, see section 3.4.

The following table shows the release plan for the web-services for MagicPED:

Component	Description	Release Date	Provider
MD RTP Wrapper	The plugin that accesses the Task Web-Services	30.04.2015	OFF
Procedure DB	Storage for procedures	30.04.2015	OFF
Execution Time Calculator	Calculates the execution times for the task hierarchy	30.06.2015	OFF
Workload Calculator	Calculates the workload for the task hierarchy	31.06.2015	OFF

3.2 CASCaS (OFF)

3.2.1 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.

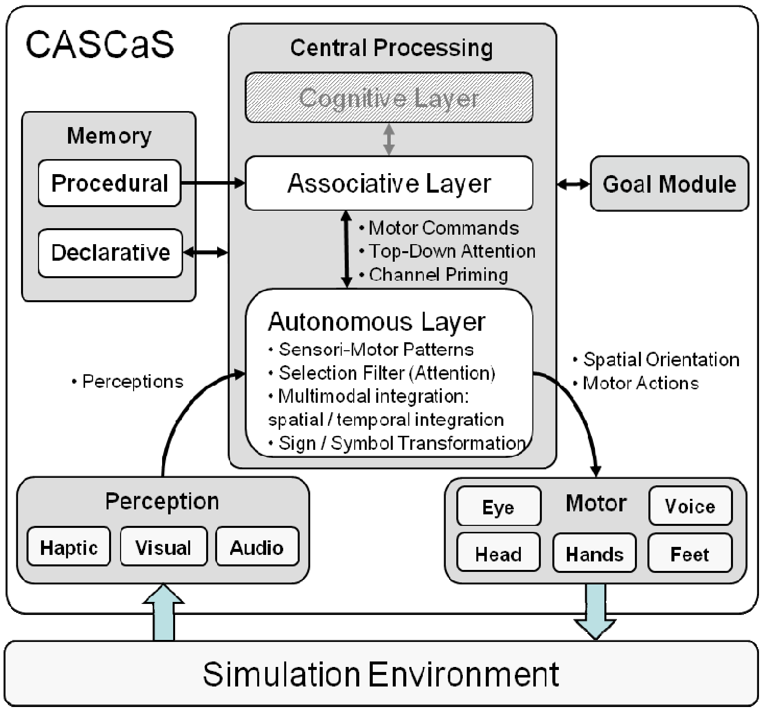


Figure 28: Structure of the cognitive architecture CASCaS with all internal components and the major data flows.

Figure 28 shows the current version of the architecture 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. 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.

3.2.2 State-of-the-Art

Today, human error is one of the main factors in transportation accidents. In aeronautics 60-80% of commercial aircraft accidents [18] and in automotive 84% of car accidents [93] 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. [87]). 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 [43], [12].

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 OFFIS, the cognitive architecture CASCaS (Cognitive Architecture for Safety Critical Task Simulation) has been developed since 2004 [65]. 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, [5], [6]), SOAR [74], [55] and MIDAS [25], [31], [33]. 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 [30], including ACT-R and (Air-)MIDAS. CASCaS has been applied in several projects, in order to model perception [63], attention allocation [98], decision making of drivers [95] and human errors of aircraft pilots [62] and car drivers [64].

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. [40], [39], [41]. This extension is currently under development.

3.2.3 MTT Description

3.2.3.1 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) OFFIS 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 OFFIS 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.

3.2.3.2 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 OFFIS 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.

3.2.3.3 AdCoS Use-Cases

In the domain of driver modelling OFFIS and TAKATA have agreed on a use case where the model should be used to simulate the interaction between the driver and the HMI developed by TAKATA. 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.

3.2.3.4 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.

3.2.4 RTP Integration Plan

Main purpose of CASCaS is the real time simulation. A dedicated framework for simulation (IEEE 1516 HLA Standard) has been chosen for interfacing CASCaS with other simulation tools. OSLC is not suitable for running real-

time simulations, e.g. it does not offer services for time management. Therefore the use of OSLC for connecting simulation is not appropriate. The surrounding tools for producing the needed input for CASCaS (e.g. MagicPED), controlling the simulation (e.g. CoSimECS) or processing the output are candidates for integration with the HF-RTP. HF-RTP integration of MagicPED, which is described in D2.4, has been started. Development of CoSimECS is currently postponed to end of second year of HoliDes.
In addition, CASCaS is integral part of the Human Efficiency Evaluator (HEE), see next section for details.

3.3 Human Efficiency Evaluator (OFF)

3.3.1 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 analysing 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 that represent the targeted audience need to be recruited and paid, which is problematic in safety-critical system domains in that specifically and extensively trained operators are needed (i.e. pilots and physician).

Further on, user testing can only be scaled to a very limited extend: Often, because of costs and time, 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
- Human Efficiency Evaluator – to model the interaction capabilities of other environments, to demonstrate procedures for common tasks and

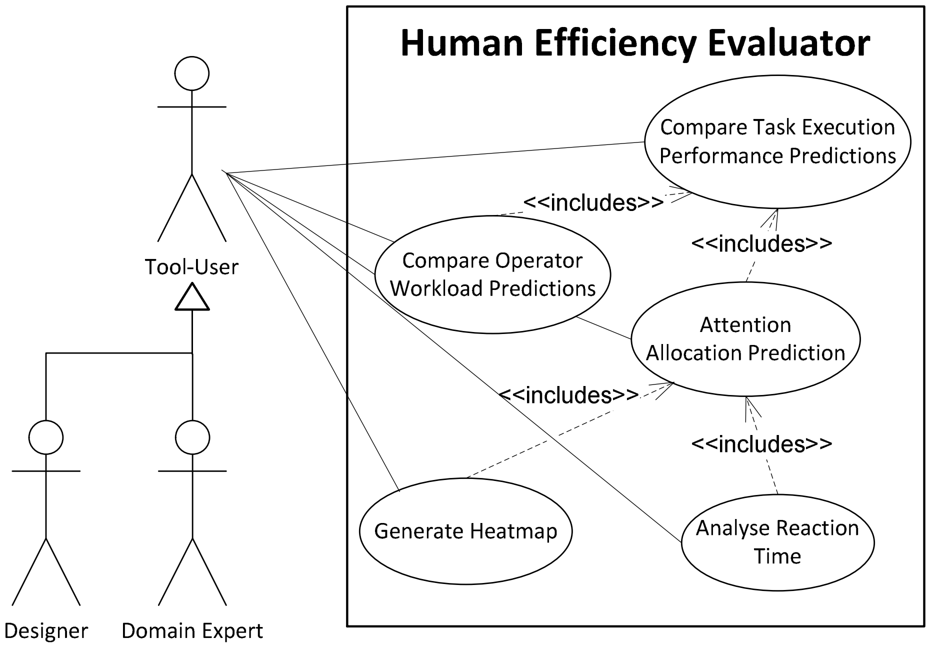

Human Efficiency Evaluator
Tool-User
Designer
Domain Expert
Compare Task Execution Performance Predictions
<<includes>>
<<includes>>
Compare Operator Workload Predictions
Attention Allocation Prediction
<<includes>>
<<includes>>
Generate Heatmap
Analyse Reaction Time

Figure 29 : UML Use Case Diagram of the Human Efficiency Evaluator

to execute

- CASCaS – a cognitive architecture for prediction of human behaviour, allowing analysis of HF Metrics

The HEE tool supports evaluation in early design phases to predict task performance, operators' workload, the attention allocation of the operator and operator's reaction times of different HMI designs by simulating the human behaviour with a cognitive architecture based on low-fidelity prototypes such as photos, screenshots or sketches as input.

It is possible to analyse and compare HMI designs:

- without the involvement of real users,
- without implementing a system prototype,
- with a huge amount of different variants in a short amount of time, and
- without the need to involve experts in user testing or cognitive analysis

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 result quality rather 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 comparative results are in focus (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 Fits 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.

3.3.2 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 model have got 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 [44] 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 [27] 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 [75] into a cognitive model by associated cognitive activity patterns (CAP). Several re-usable CAPs have been implemented to generate declarative and procedural ACT-R [5] structures e.g. to describe scanning, observation, monitoring or action execution of an operator. With CAP HTAmap implements a concept for re-using concepts within a cognitive model that is task-centric while the HEE implements re-usability on an instrument level by linking instrument designs to cognitive models.

The Automation Design Advisor Tool (ADAT) [88] 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. Bothe tools evaluate the design using evaluations scales. ADAT focuses on graphical design evaluation while the HEE invests in task and workload predictions.

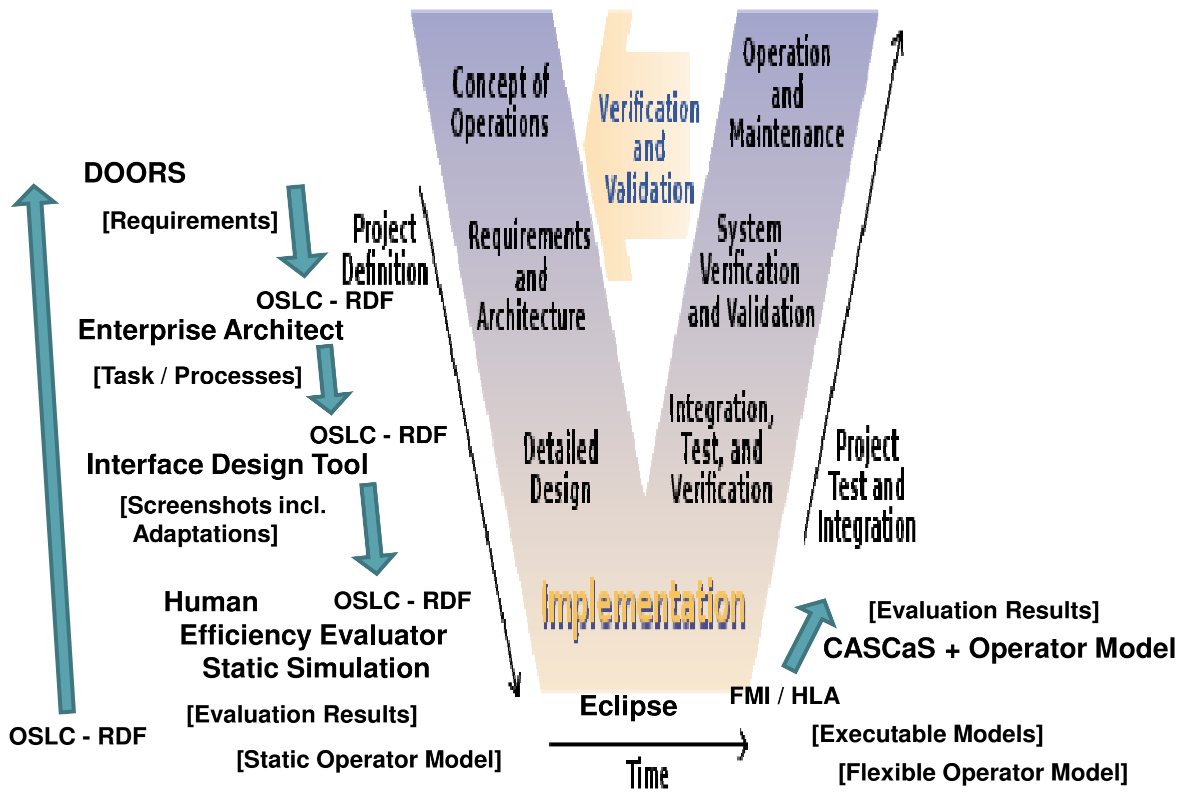


Figure 30: Human Efficiency Evaluator as part of a development process based on the V-Model.

COGENT [24] 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." (Taken from "Easing the Cognitive Analysis of HMI Designs - The Human Efficiency Evaluator" by Feuerstack et al., currently under review).

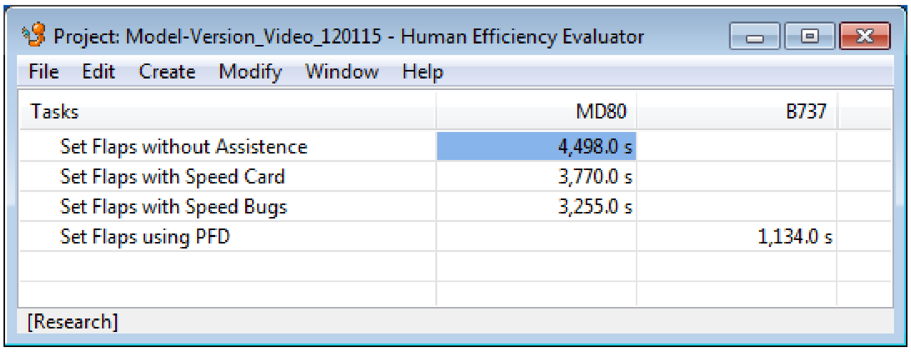
3.3.3 MTT Description

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.

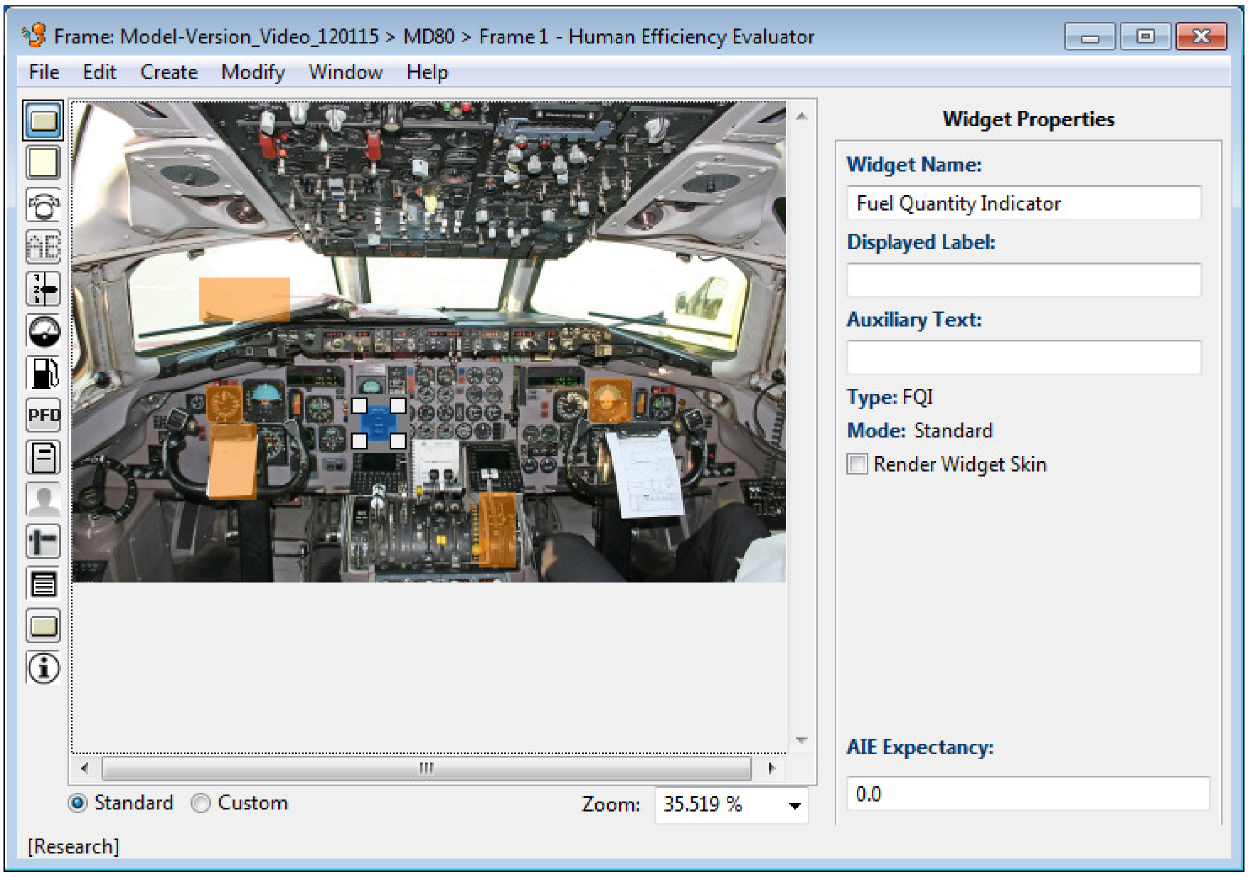
But there are several fundamental differences between both: CogTool focuses on performance prediction of WIMP (Windows, Icons, Menus, Pointer) interfaces. Therefore, the annotation of design sketches is based on a fixed palette of the most common WIMP widgets like buttons, menu bars, and radio buttons. 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.

Therefore, we re-implemented the entire backend of CogTool to use our cognitive architecture CASCaS and integrated support for generating operator models. Further on, 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


Project: Model-Version_Video_120115 - Human Efficiency Evaluator
File
Edit
Create
Modify
Window
Help
[Research]

Tasks	MD80	B737
Set Flaps without Assistence	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

Figure 31: Project Overview


Frame: Model-Version_Video_120115 > MD80 > Frame 1 - Human Efficiency Evaluator
File
Edit
Create
Modify
Window
Help
PFD
Standard
Custom
Zoom:
[Research]

Widget Properties

Widget Name: Fuel Quantity Indicator

Displayed Label:

Auxiliary Text:

Type: FQI

Mode: Standard

☐ Render Widget Skin

AIE Expectancy: 0.0

Zoom: 35.519 %

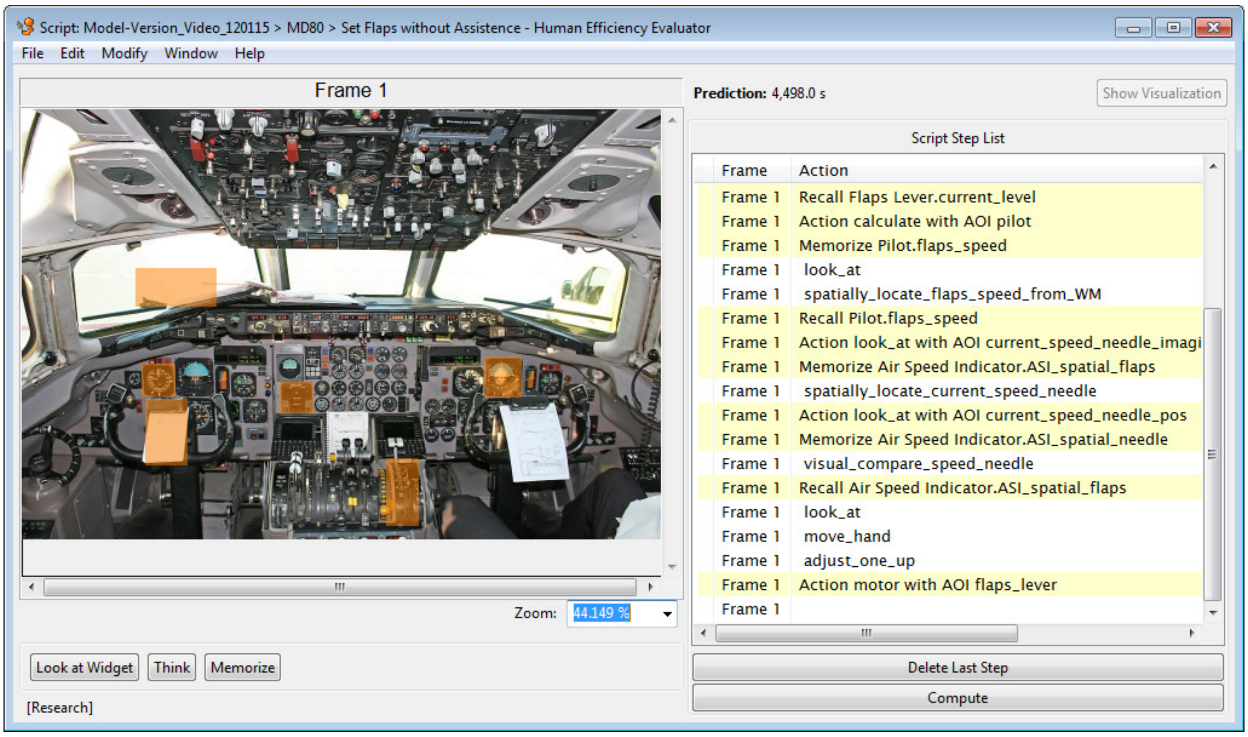
Figure 32: HEE design annotation view.

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).

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 31**). 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 accessed by clicking on the column name of a design. The design editor is depicted by **Figure 32**. A palette at the left side



Frame	Action
Frame 1	Recall Flaps Lever.current_level
Frame 1	Action calculate with AOI pilot
Frame 1	Memorize Pilot.flaps_speed
Frame 1	look_at
Frame 1	spatially_locate_flaps_speed_from_WM
Frame 1	Recall Pilot.flaps_speed
Frame 1	Action look_at with AOI current_speed_needle_imagi
Frame 1	Memorize Air Speed Indicator.ASI_spatial_flaps
Frame 1	spatially_locate_current_speed_needle
Frame 1	Action look_at with AOI current_speed_needle_pos
Frame 1	Memorize Air Speed Indicator.ASI_spatial_needle
Frame 1	visual_compare_speed_needle
Frame 1	Recall Air Speed Indicator.ASI_spatial_flaps
Frame 1	look_at
Frame 1	move_hand
Frame 1	adjust_one_up
Frame 1	Action motor with AOI flaps_lever
Frame 1	

Figure 33: HEE Procedure Demonstration.

shows all annotation options, with a set of pre-defined widgets and instruments for the specific application domain. After the operator is "virtually" positioned relatively 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 33**.

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 33**. 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 (c.f. **Figure 33**). The task performance time prediction is the shown above the scrip (c.f. **Figure 33**) and as well added to the project overview (c.f. **Figure 31**) to ease the comparison between different design variants.

3.3.4 RTP Integration Plan

Figure 30 illustrates how the HEE integrates in the V-model process: Based on an initial requirements analysis and an optional task design the HEE tool requires screenshots of the interface and instrument or display designs as the basic input. The results are predictions based on metrics that support the initial decision for one interface design. Different to the HEE CASCaS (c.f. section 3.2) is usually applied to HMI prototypes that already passed the implementation phase of the V-model. Evaluating a functional prototype with CASCaS offers an improved prediction quality while requiring more effort in the operator model programming. Also a programmatically link between the CASCaS and the HMI needs to be implemented, so that the operator model can access the HMI prototype.

The HEE is thought to interconnect with a task modelling tool and a user interface mock-up tool to retrieve its initial input. The generated resource models as well as the prediction results are available as outputs to be offered to other tools. Currently, common meta-models are being defined to drive this interconnection.

The following table shows the release plan for the HEE for the upcoming modelling with the Health and the Control Room AdCos:

Component	Description	Release Date	Provider
HEE 1.9.3	Initial version to start basic modelling of the 3D acquisition use case	13.02.2015	OFF
HEE 2.0	Improved version to model the entire 3D acquisition use case.	14.08.2015	OFF
HEE 2.2	Improved version for initial modelling of border use case	31.01.2016	OFF
HEE 2.4	Final version with entire model for border control use case.	31.06.2016	OFF

Table Human Efficiency Evaluator planned updates.

3.4 TrainingManager (OFF, TRS)

3.4.1 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".

3.4.2 State-of-the-Art

There are several existing professional training management tools, like MINT[6], prodefis COURSE[7], SkyManager[8], or ETA (Education & Training Administration)[9], 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.

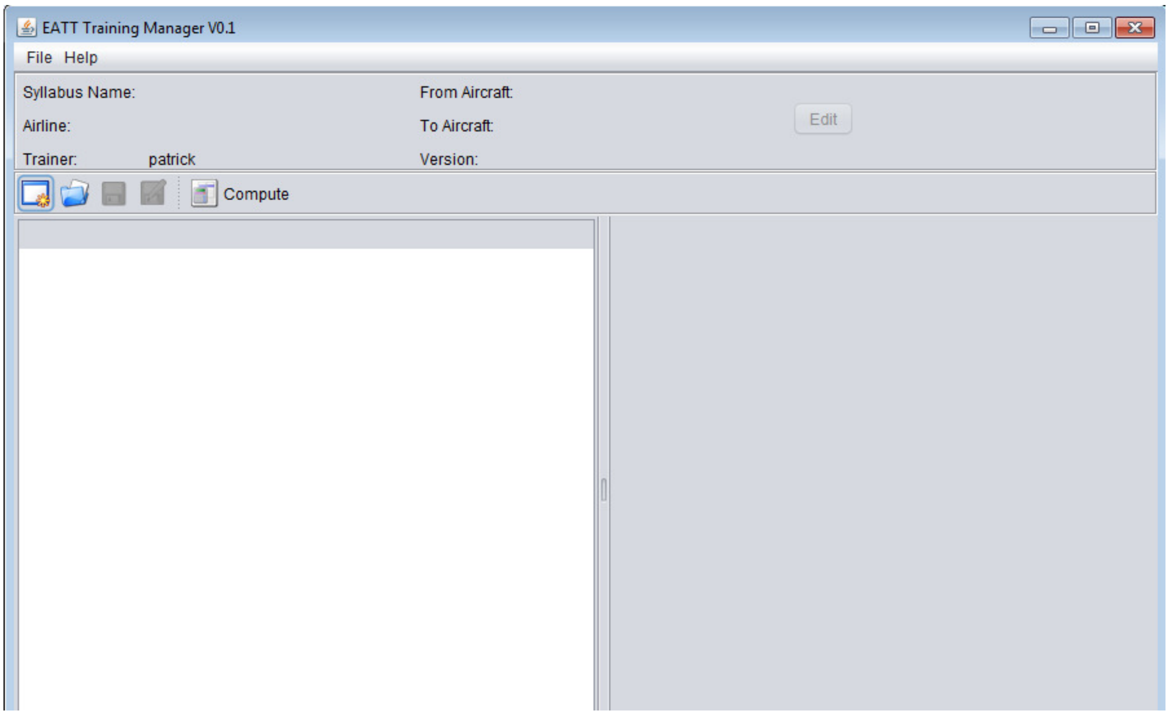
Currently, transition training is not adapted, i.e. the trainees follow the same procedure for licensing, then pilots without previous experience. Thus, the effort taken within HoliDes is completely new, and must also undergo some kind of certification by authorities at a later stage.

3.4.3 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 34. 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 35. 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 adaptions 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.

[6] http://www.media-interactive.de/
[7] http://www.prodefis.de/aviation/de/products/prodefis-course.html
[8] http://www.skymanager.com/
[9] http://www.talon-systems.com/

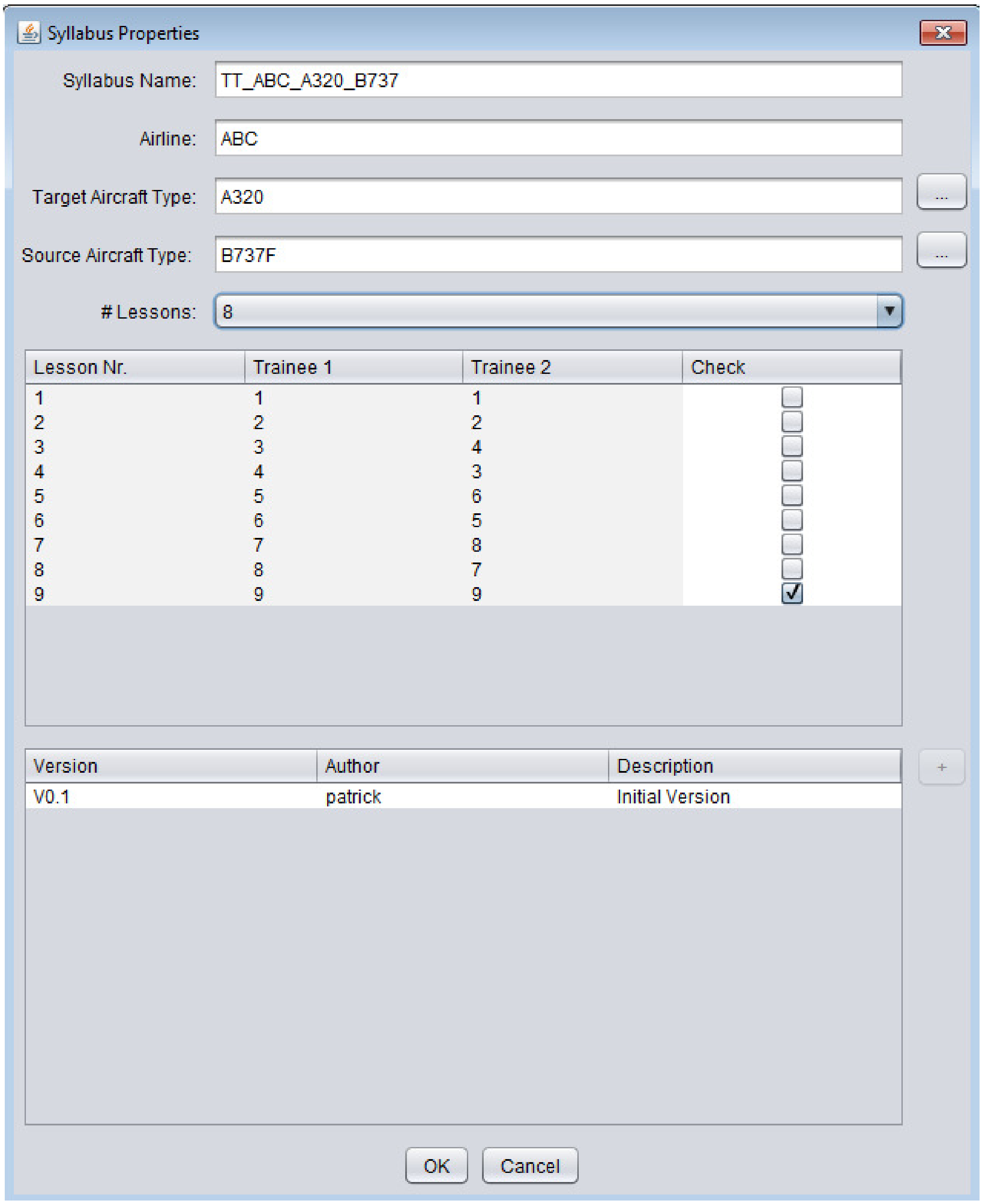


Syllabus Name:
Airline:
Trainer: patrick
From Aircraft:
To Aircraft:
Version:

Figure 34: EATT Training Manager - Main Window

After the empty syllabus has been created (Figure 38), the trainer can start in a first step to assign content to the lessons. As explained in section 2.1.6 and depicted in Figure 37, 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_COCKPIT_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.



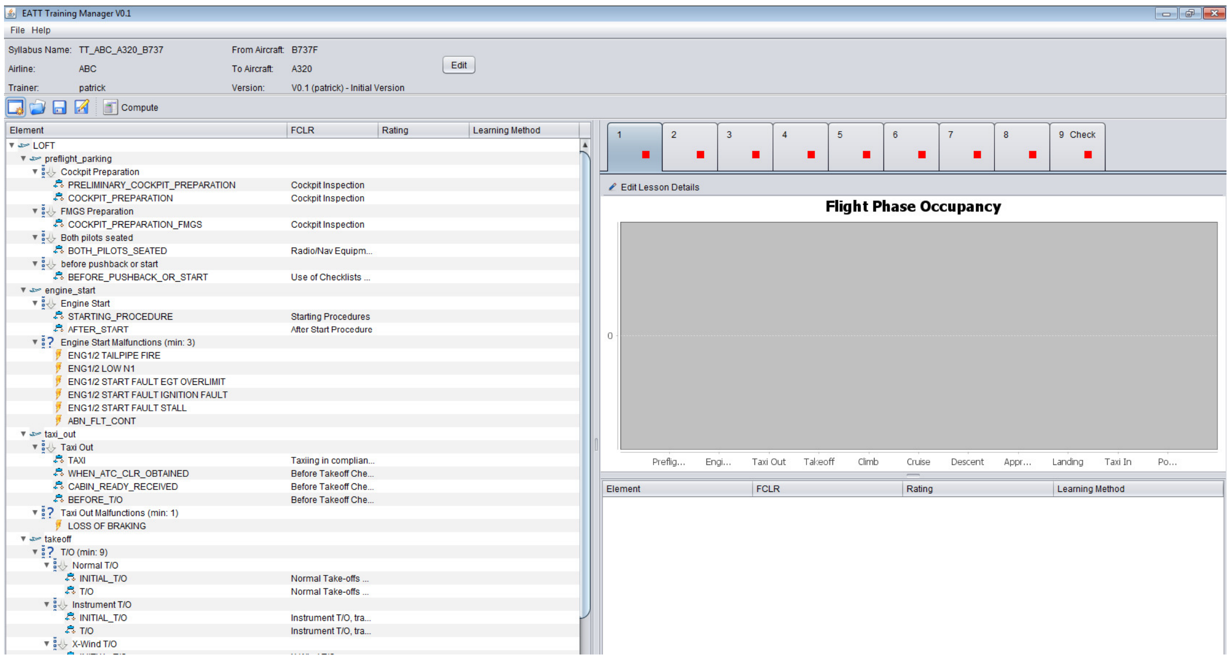
Syllabus Properties

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	☐
2	2	2	☐
3	3	4	☐
4	4	3	☐
5	5	6	☐
6	6	5	☐
7	7	8	☐
8	8	7	☐
9	9	9	✓

Version	Author	Description
V0.1	patrick	Initial Version

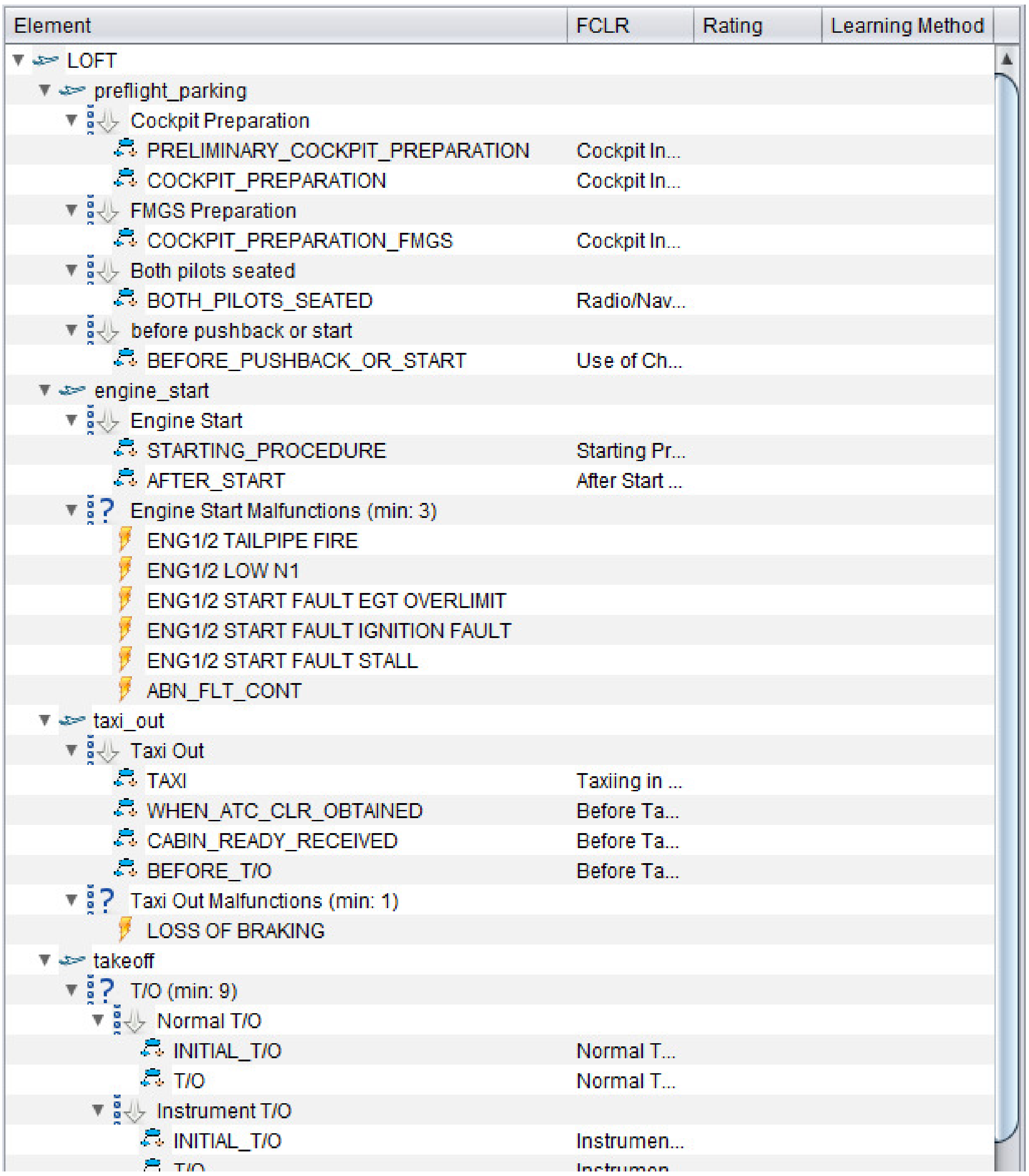
Figure 35: EATT TrainingManager - Syllabus Wizard



Syllabus Name:	TT_ABC_A320_B737	From Aircraft:	B737F
Airline:	ABC	To Aircraft:	A320
Trainer:	patrick	Version:	V0.1 (patrick) - Initial Version

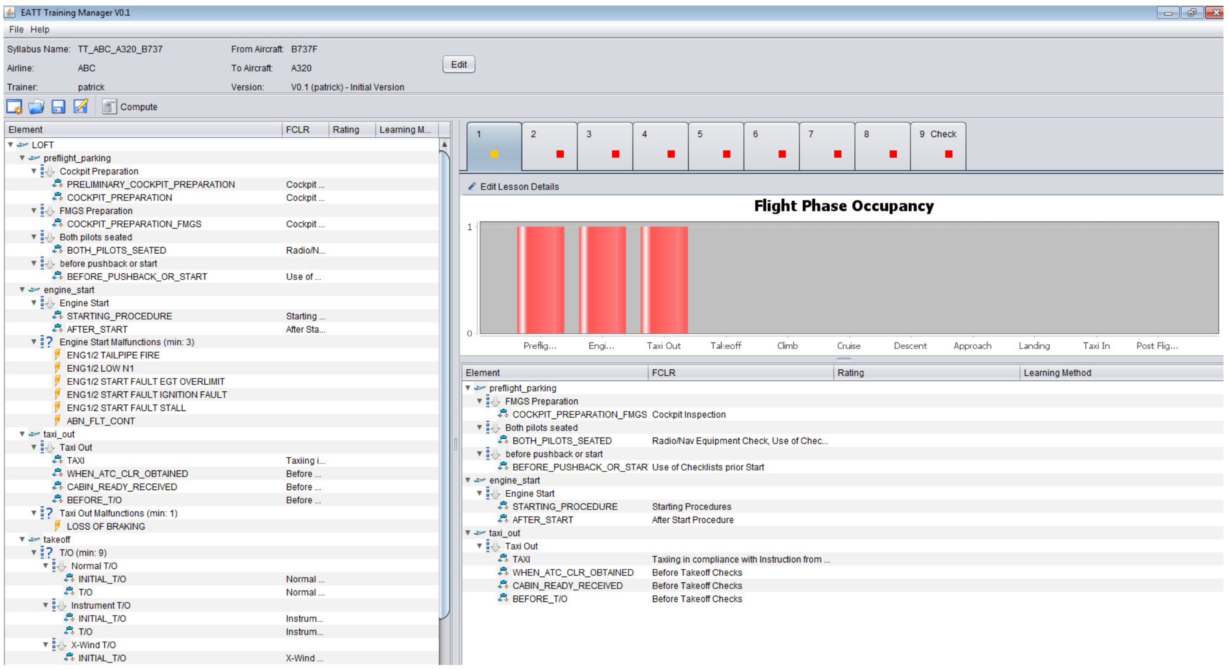
Element	FCLR	Rating	Learning Method
LOFT			
preflight_parking			
Cockpit Preparation			
PRELIMINARY_COCKPIT_PREPARATION	Cockpit Inspection		
COCKPIT_PREPARATION	Cockpit Inspection		
FMGS Preparation			
COCKPIT_PREPARATION_FMGS	Cockpit Inspection		
Both pilots seated			
BOTH_PILOTS_SEATED	Radio/Nav Equipm...		
before pushback or start			
BEFORE_PUSHBACK_OR_START	Use of Checklists ...		
engine_start			
Engine Start			
STARTING_PROCEDURE	Starting Procedures		
AFTER_START	After Start Procedure		
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 complian...		
WHEN_ATC_CLR_OBTAINED	Before Takeoff Che...		
CABIN_READY_RECEIVED	Before Takeoff Che...		
BEFORE_T/O	Before Takeoff Che...		
Taxi Out Malfunctions (min: 1)			
LOSS OF BRAKING			
takeoff			
T/O (min: 9)			
Normal T/O			
INITIAL_T/O	Normal Take-offs ...		
T/O	Normal Take-offs ...		
Instrument T/O			
INITIAL_T/O	Instrument T/O, tra...		
T/O	Instrument T/O, tra...		
X-Wind T/O			

Figure 36: EATT TrainingManager - Step 1



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 37: Tree with Lesson Entries



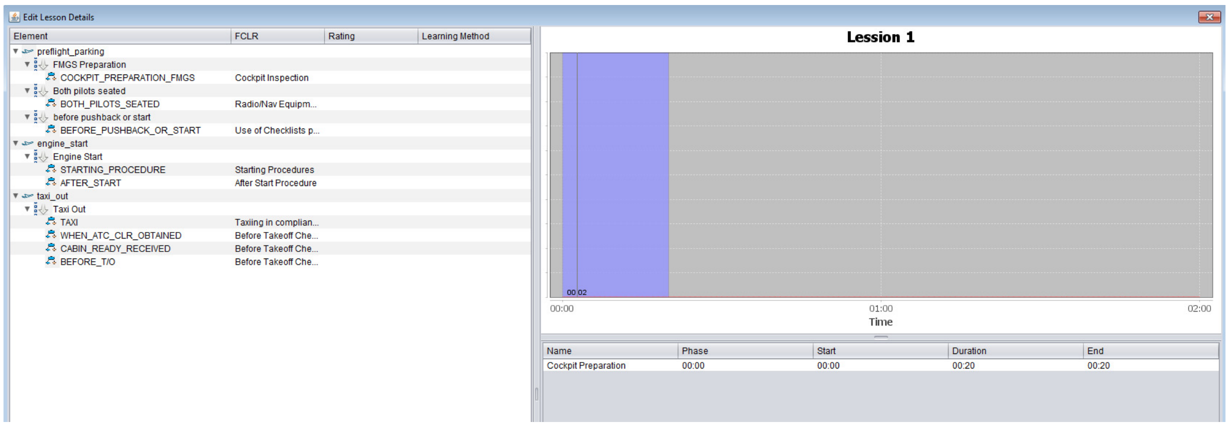
Syllabus Name:	TT_ABC_A320_B737
Airline:	ABC
Trainer:	patrick
From Aircraft:	B737F
To Aircraft:	A320
Version:	V0.1 (patrick) - Initial Version

Element	FCLR	Rating	Learning M...
LOFT			
preflight_parking			
Cockpit Preparation			
PRELIMINARY_COCKPIT_PREPARATION	Cockpit ...		
COCKPIT_PREPARATION	Cockpit ...		
FMGS Preparation			
COCKPIT_PREPARATION_FMGS	Cockpit ...		
Both pilots seated			
BOTH_PILOTS_SEATED	Radio/N...		
before pushback or start			
BEFORE_PUSHBACK_OR_START	Use of ...		
engine_start			
Engine Start			
STARTING_PROCEDURE	Starting ...		
AFTER_START	After Sta...		
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 i...		
WHEN_ATC_CLR_OBTAINED	Before ...		
CABIN_READY_RECEIVED	Before ...		
BEFORE_T/O	Before ...		
Taxi Out Malfunctions (min: 1)			
LOSS OF BRAKING			
takeoff			
T/O (min: 9)			
Normal T/O			
INITIAL_T/O	Normal ...		
T/O	Normal ...		
Instrument T/O			
INITIAL_T/O	Instrum...		
T/O	Instrum...		
X-Wind T/O			
INITIAL_T/O	X-Wind ...		

Element	FCLR	Rating	Learning Method
preflight_parking			
FMGS Preparation			
COCKPIT_PREPARATION_FMGS	Cockpit Inspection		
Both pilots seated			
BOTH_PILOTS_SEATED	Radio/Nav Equipment Check, Use of Chec...		
before pushback or start			
BEFORE_PUSHBACK_OR_STAR	Use of Checklists prior Start		
engine_start			
Engine Start			
STARTING_PROCEDURE	Starting Procedures		
AFTER_START	After Start Procedure		
taxi_out			
Taxi Out			
TAXI	Taxiing in compliance with Instruction from ...		
WHEN_ATC_CLR_OBTAINED	Before Takeoff Checks		
CABIN_READY_RECEIVED	Before Takeoff Checks		
BEFORE_T/O	Before Takeoff Checks		

Figure 38: EATT TrainingManager - Step 1 with assigned FCLRs

After the requirements (flight crew licensing requirements; FCLR) have been allocated to the sessions (see Figure 38), the trainer can start the second step, the fine planning of each session, see also Figure 39, where all content that has been assigned to the session is assigned on a timeline. Also here, consistency checks are applied.



Element	FCLR	Rating	Learning Method
preflight_parking			
FMGS Preparation			
COCKPIT_PREPARATION_FMGS	Cockpit Inspection		
Both pilots seated			
BOTH_PILOTS_SEATED	Radio/Nav Equipm...		
before pushback or start			
BEFORE_PUSHBACK_OR_START	Use of Checklists p...		
engine_start			
Engine Start			
STARTING_PROCEDURE	Starting Procedures		
AFTER_START	After Start Procedure		
taxi_out			
Taxi Out			
TAXI	Taxiing in complian...		
WHEN_ATC_CLR_OBTAINED	Before Takeoff Che...		
CABIN_READY_RECEIVED	Before Takeoff Che...		
BEFORE_T/O	Before Takeoff Che...		

Name	Phase	Start	Duration	End
Cockpit Preparation	00:00	00:00	00:20	00:20

Figure 39: EATT TrainingManager - Step 2

3.4.4 RTP Integration Plan

Figure 40: EATT Training Manger - Architecture

As described in the section on MagicPED, additional algorithms for the TrainingManager are planned to be implemented as Web-Services, accessible via the HF-RTP. The TrainingManager will have a component, based on the eclipse LYO library[10], for implementing the OSLC. The additional web-services will:

- Calculate the similarity of two procedures.
- Calculate an advice which training/learning strategy is best for the procedure.

The following table shows the release plan for the web-services for the TrainingManager:

Component	Description	Release Date	Provider
TrainingManager RTP Wrapper	The plugin that accesses the Task Web-Services	30.04.2015	OFF
Procedure DB	Storage for procedures	30.04.2015	OFF
Procedure Comparator	Calculates the similarity of procedures	30.04.2015	OFF
Training Advisor	Calculates best learning strategy	31.06.2015	OFF

3.5 COSMODRIVE and COSMO-SIVIC (IFS)

3.5.1 Introduction

COSMODRIVE is a Cognitive Simulation Model of the car Driver developed at IFSTTAR, in order to provide computational simulation of car drivers. The general objective is to virtually simulate the human drivers' perceptive and

[10] http://www.eclipse.org/lyo/

cognitive activities implemented when driving a car, through an iterative "Perception-Cognition-Action" regulation loop (Figure 41).

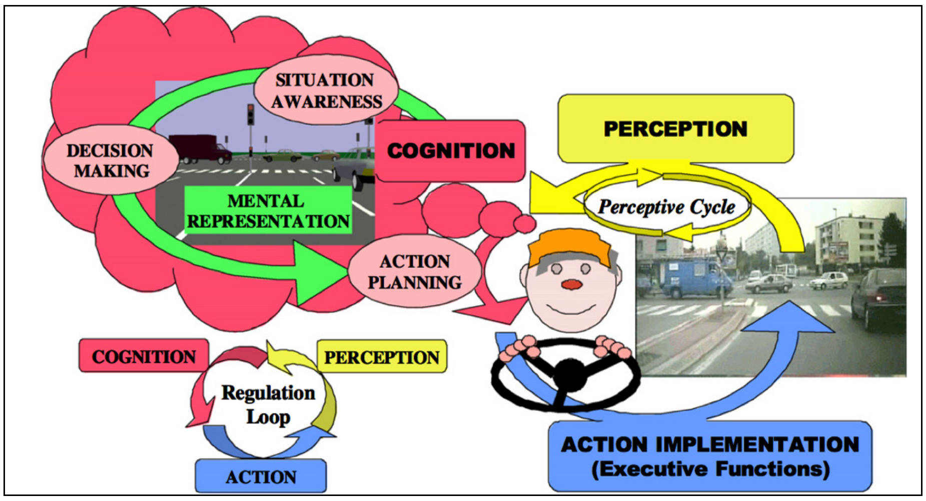


Figure 41: Overview of COSMODRIVE Model theoretical approach

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"; [13]) 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; [14]) 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 State-of-the-Art

COSMODRIVE is a long term research programme of IFSTTAR dedicated to cognitive simulation of human drivers [13], [14], [15]. 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 (2nd 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).

3.5.3 MTT Description

The functional architecture of the new version of the COSMODRIVE model specifically implemented for HoliDes is composed of three main modules (Figure 42): 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.

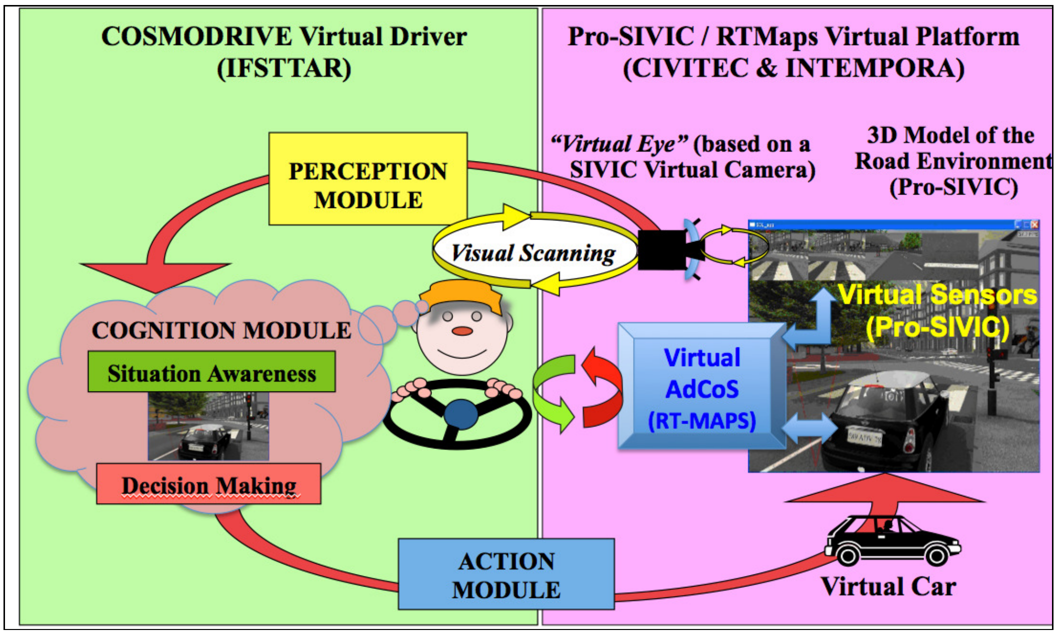

COSMODRIVE Virtual Driver (IFSTTAR)
Pro-SIVIC / RTMaps Virtual Platform (CIVITEC & INTEMPORA)
PERCEPTION MODULE
"Virtual Eye" (based on a SIVIC Virtual Camera)
3D Model of the Road Environment (Pro-SIVIC)
Visual Scanning
COGNITION MODULE
Situation Awareness
Decision Making
Virtual Sensors (Pro-SIVIC)
Virtual AdCoS (RT-MAPS)
ACTION MODULE
Virtual Car

Figure 42: 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.

The Driving Scenarios and MOVIDA-AdCoS Use Cases to be investigated by IFSTTAR will concern driving situation occurring on a two-lanes Inter-Urban Highway limited to 90 km/h (Figure 43).

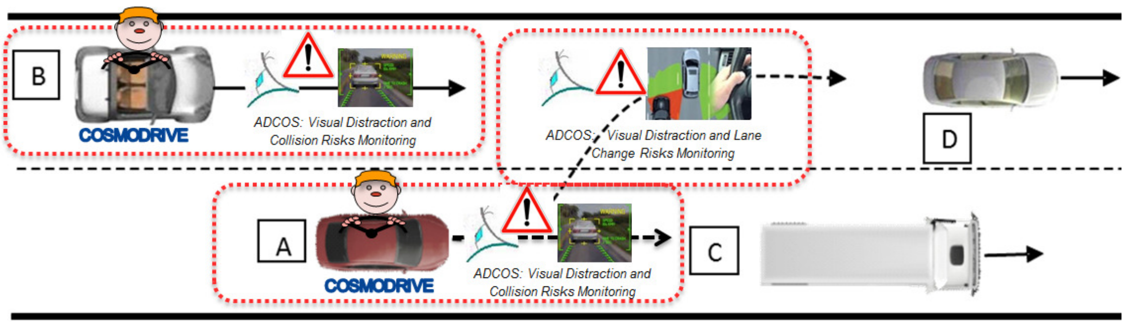

B
COSMODRIVE
ADCOS: Visual Distraction and Collision Risks Monitoring
ADCOS: Visual Distraction and Lane Change Risks Monitoring
D
A
COSMODRIVE
ADCOS: Visual Distraction and Collision Risks Monitoring
C

Figure 43: Driving Scenarios and Use cases for ADCOS based on MOVIDA functions, to be designed and evaluated with COSMODRIVE by IFS in WP9

In this driving context, the MOVIDA-AdCoS will be designed in order to support drivers in Car A and/or in Car B. Regarding car A drivers, the aim will be to assist them in an adaptive way in case of critical visual distracted while approaching a slower vehicle (vehicle C), by managing the collision with it and/or by supporting a Lane Change manoeuvre (overtaking). For Car B drivers, this AdCoS will be mainly in charge to support collision risk in case of critical lane change of Car A, more particularly if this lane change occurs when car B driver is visually distracted.

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 44), 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; [73]), 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.

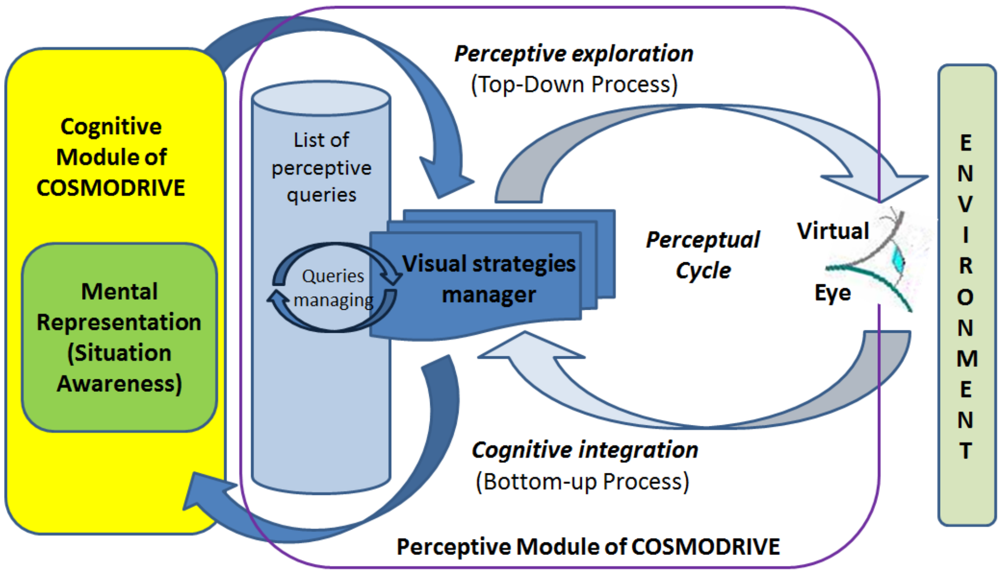

Cognitive Module of COSMODRIVE
Mental Representation (Situation Awareness)
List of perceptive queries
Queries managing
Visual strategies manager
Perceptive exploration
(Top-Down Process)
Perceptual Cycle
Virtual
Eye
Cognitive integration
(Bottom-up Process)
Perceptive Module of COSMODRIVE
ENVIRONMENT

Figure 44: 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).

3.5.4 RTP Integration Plan

COSMODRIVE integration as the HF component of a tailored HF-RTP for automotive domain is a joint effort of IFSTTAR, INTEMPORA and CIVITEC, currently implemented in WP4 and WP9 (complementary descriptions of this integrative work are also presented in D4.4 and D9.4).

3.5.4.1 Towards a tailored HF-RTP for automotive domain

The following figure (Figure 45) provides an overview of the current architecture of this integrated tool chain – as a Virtual Human Centred Design platform (V-HCD) - specifically designed for HoliDes objectives of virtual AdCoS design and evaluation. In this platform, RT-Maps plays a key role for connecting the different MTT, supporting ADAS and AdCoS simulation, and allowing COSMODRIVE to perceive the road environment in Pro-SIVIC, to drive the virtual car, and to interact with the AdCoS.

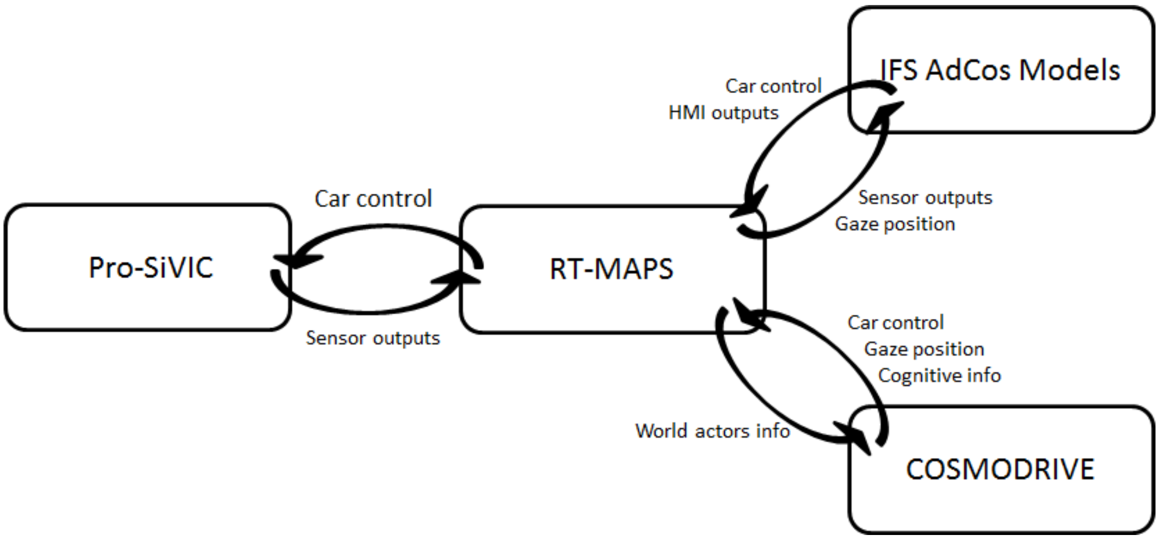


Figure 45: RTMaps-based platform for V-HCD of AdCoS with COSMODRIVE

3.5.4.2 COSMODRIVE implementation for an HF-RTP tailoring in WP9

COSMODRIVE implementation principle in this V-HCD platform (as a particular example of a tailored HF-RTP in WP9; D9.4) is based on previous IFS experience in model implementation and by considering the specific aims and challenges investigated in HoliDes project. To better support COSMODRIVE future use in HoliDes and its interoperability with other MTTs, several principles were applied in this new computational version of the model:

- **Distributed computing**: In its complete description, the COSMODRIVE requires much computing power. For instance, the need of multiple 3D representations, each having multiple algorithms with various levels of complexity. Or anticipation, which even though it's a relatively "easy" human task, is incredibly more difficult and time consuming for a computer. To cope with this, the model can be distributed over multiple computers on the same network.
- **High interoperability**: Given the amount of tools providers for the HoliDes project, the implementation is to be able to interact with as many of them as possible.
- **Customization**: Both on software and model level, the implementation need to offer configuration parameters to allow the end user to optimize the model for its use.
- **Human-Machine Interface for model developers**: The implementation provides deep visual feedback to the state of each module and process of COSMODRIVE, as opposed to only providing outputs of the whole model.
- **Data exchange (export/import)**: All the data produced by the COSMODRIVE's modules and processes can be exported to be analysed either at runtime or at a later time. This data can also be used to (re)start the model at a given state of the simulation.

3.5.4.3 Design methods and technical tools:

To fulfil the Drivers cognitive simulation requirements in HoliDes, two main issues needs answer: the **3D modelling** needed for mental representations simulation and the multiple **communications** channels between them and with the outside.

3.5.4.3.1 Mental models simulation with 3D engine

To simulate drivers' visual-spatial mental models, it was decided to use the open source Massive G Engine[11] (MGEngine). This choice was made for two main reasons:

- CIVITEC uses this engine as the core of Pro-SiVIC, which allows great interoperability between our software.
- Previous work has been done at IFSTTAR with MGEngine and provided great results and knowledge of the software.

MGEngine is written in C++ and is Windows and Linux compatible. Its main feature is the ability to dynamically load any compliant shared library with minimal amount of work needed. It also offers an easy to use scripting language which allows anyone to configure the engine and its components.

This fits well with the requirements, especially since COSMODRIVE can be divided in various modules and processes, which can each be represented as an MGEngine component (shared library). Multiple instances of MGEngine can then be used, each one only loading the components it needs, as specified by in script files written by the user.

3.5.4.3.2 Communication

All COSMODRIVE's communications are handled by ZeroMQ (ZMQ: http://zeromq.org/). This library allows simple and fast machine to machine (m2m) communication. It supports multiple messaging patterns which are used in Cosmodrive (Pub-Sub, Push-Pull) and carries messages over various protocols (TCP, inproc, etc.).

Since latency between modules can quickly become an issue in COSMODRIVE, the ability to select the underlying protocol is very useful. This library also is open source, offers supports for many programming languages and has a large and active community ZMQ and MGEngine provide a strong core for COSMODRIVE's implementation. Using this combination, a new MGEngine component can quickly be realized and plugged-in with the rest of the model. This allows for quick iterations development processes, as well as multiple various implementations of the same modules/process.

[11] MGEngine: https://sourceforge.net/projects/mgengine/

3.5.4.3.3 Illustration

The following figure is a UML Component Diagram illustrating COSMODRIVE’s Virtual Eye. In this Figure 46, each sub-component (“Peripheral vision”, “Foveal vision”) is a **process** is terms of COSMODRIVE as a model, i.e. it will take inputs and provide outputs. In terms of the software, those are **components**, meaning that they each are a unique entity (shared library) that can be independently loaded and configured by the MGEngine. The inputs and outputs mentioned in this diagram are handled in the software using ZMQ.

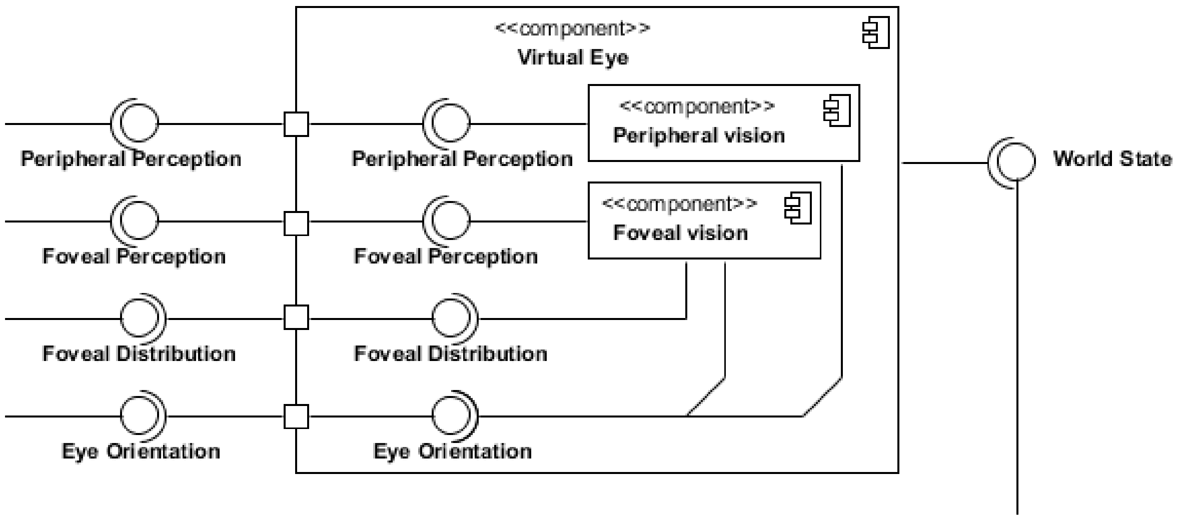


Figure 46: UML Component Diagram of COSMODRIVE’s Virtual Eye

3.5.4.4 COSMODRIVE model Inputs/Outputs

Like a human driver model, COSMODRIVE mainly has the same interface with the world as a human has. For input, the model needs a world to move in, and for output, it will provide human control of the car. Other possible outputs will be discussed later.

3.5.4.4.1 The Road Environment

COSMODRIVE, as any human driver, needs a world (i.e. road environment) in which the car will move. Theoretically, our actual real world could be used, but at the current state of development, only simulated world are supported.

This simulated world needs to provide a way to control a car remotely (i.e. from COSMODRIVE), and needs to share information about the state of the world with COSMODRIVE. Indeed, COSMODRIVE's main entry point is the perception module, most importantly the Virtual Eye. To perceive, the Virtual Eye needs to be in direct interaction with the world. From a software standpoint, the world is a 3D engine, and each simulator uses a different 3D engine. Thus, making the Virtual Eye compatible with every simulator would require tremendous work. To solve this issue, we recreate the simulation inside a special dedicated component of COSMODRIVE, where the Virtual Eye can perceive in our own 3D engine (MGEngine). This "slave" simulation then synchronized with the "master" one. At the moment, only position and orientation of the actors are synchronized, but more options are considered (e.g. tail lights) depending on the world's software. This solution requires some preliminary work to port 3D assets and coordinates system from the master engine to COSMODRIVE's, but the current work with Pro-SiVIC has been facilitate as the use the same core engine (MGEngine) as COSMODRIVE.

3.5.4.4.2 Car Control

COSMODRIVE provides the (simulated) world with commands to control the car. Currently, it is composed of:

- A normalized steering wheel value, with -1 being the maximum left and +1 the maximum right.
- A normalized acceleration value, with -1 being minimum acceleration (i.e. maximum deceleration) and +1 the maximum acceleration.

Gearbox currently is not supported, but pedal depression (instead of acceleration) might be in the near future (depending on whether a simulator with such features is provided).

3.5.4.4.3 Gaze Orientation

Even though the Virtual Eye is internal to COSMODRIVE, the gaze orientation is provided as an output (i.e. expected to be in the future in a similar way of a measures collected among real human provides by eye-tracking systems).

3.5.4.4.4 Internal States of the model

The internal state of each module, process and representation can be provided as a runtime output. There currently is no formal specification of those outputs, but it is being worked on and will be discussed with HoliDes partners.

3.5.4.4.5 Model Configuration

The model can be configured before launch on different level. First, using MGEngine scripting files, the user can choose which components to use (a default configuration is provided), how to distribute them and most importantly each component's configuration. Indeed, they will each have specific variable that have an impact on the processes, which can be seen has the driver's skills, cognitive capabilities, etc. Given the nature of the model, those variables are not as explicit as those mentioned above, but default configuration will also be provided to match a specific class of driver.

3.5.4.5 COSMODRIVE interaction with other MTTs

In its current status, COSMODRIVE is interconnected with RTMaps, that is a very efficient tool to support dynamic interaction of this HF Model with other MTTs developped in HoliDes project. Indeed, RTMaps offers several way to connect various tools and share empirical data, and several HoliDes partners already use it (see typical examples in D4.4). In addition, COSMODRIVE interfacing with OSLC will be also considered for Milestone 2. However, OSLC approach seems not really adapted for dynamic and real time simulations of AdCoS use, that is of prior importance to support virtual human centred design and evaluation of AdCoS based on HF simulation models like COSMODRIVE (see D4.4 for a detailed discussion of this issue). That's why COSMODRIVE and RTMaps interconnections were developed since the beginning of the project.

The Figure 47 provides an overview of the RTMaps diagram supporting COSMODRIVE-based simulation tool chain, liable to be also connected with other MTT (like PRO-SIVIC, in this case). Some of these components are currently under development, but the main ones are already working, allowing to dynamically exchange data between the different RTMaps modules and then with other MTT. In this RTMaps diagram, the left part is responsible for getting both cars position and orientation, and sending to

COSMODRIVE. It will then synchronize its duplicate of the world. The right part shows a COSMODRIVE component providing "Car Control" (acceleration and steering) to Pro-SiVIC.

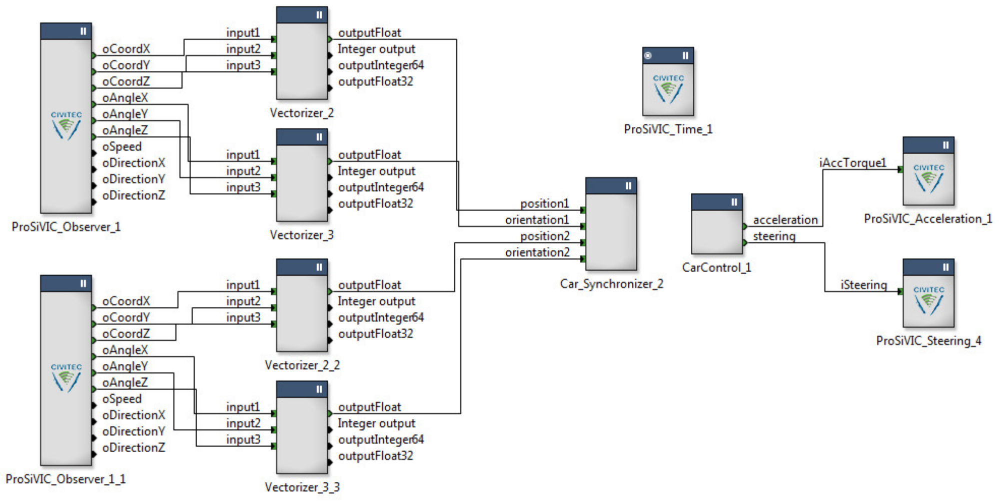
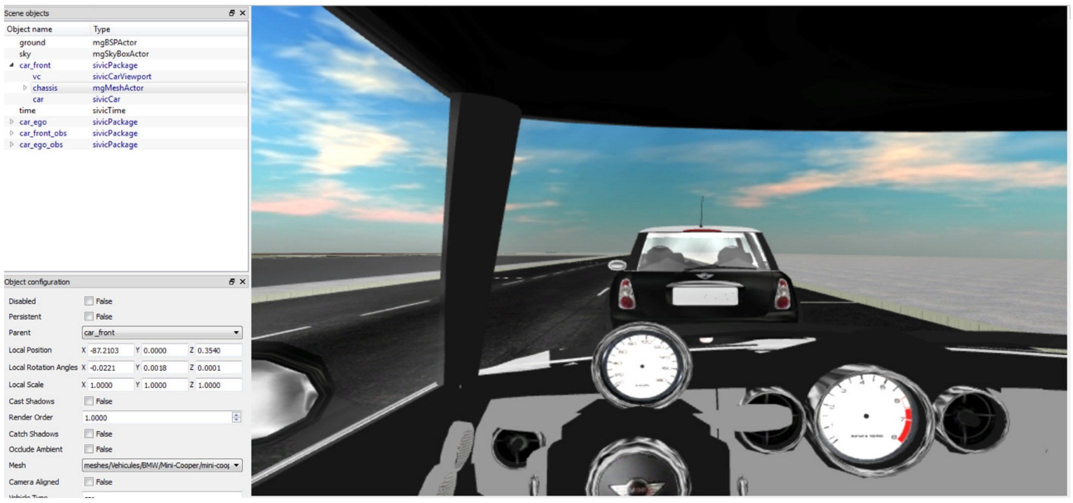


Figure 47: RTMaps diagram for Pro-SiVIC/COSMODRIVE simulation

Even though everything is still "work in progress", we would like to provide here an example illustrating the interactions between Pro-SiVIC, RTMaps and COSMODRIVE. In this frame, COSMODRIVE is in charge to simulate human drivers' perception (from visual scanning to perceptive information integration), cognitive processes (like Situation Awareness, Decision Making or action planning) and driving behaviour in a car following task.



Scene objects

Object name	Type
ground	mgBSPActor
sky	mgSkyBoxActor
car_front	sivicPackage
vc	sivicCarViewport
chassis	mgMeshActor
car	sivicCar
time	sivicTime
car_ego	sivicPackage
car_front_obs	sivicPackage
car_ego_obs	sivicPackage

Object configuration

Disabled	False		
Persistent	False		
Parent	car_front		
Local Position	X -87.2103	Y 0.0000	Z 0.3540
Local Rotation Angles	X -0.0221	Y 0.0018	Z 0.0001
Local Scale	X 1.0000	Y 1.0000	Z 1.0000
Cast Shadows	False		
Render Order	1.0000		
Catch Shadows	False		
Occlude Ambient	False		
Mesh	meshes/Vehicules/BMW/Mini-Cooper/mini-coo		
Camera Aligned	False		

Figure 48: Pro-SiVIC simulation with COSMODRIVE at the wheel

Figure 48 shows the simulated road environment and the virtual cars (ego car and lead car) simulated with Pro-SiVIC. This view is provided by a virtual camera placed at the driver's head position, and the task of COSMODRIVE is to drive the ego car and to safely follow the black car in front.

Figure 49 provides an overview of COSMODRIVE simulation results with the COSMODRIVE-RTMaps-ProSIVIC tool chain. The top-right view corresponds to the external road environment simulated with Pro-SIVIC, from the car driver's point of view. The top left view shows COSMODRIVE's Perceptive Representation of this road environment, as perceived from the Virtual Eye (the red square section corresponding to the foveal vision of this virtual eye). The bottom left view corresponds to COSMODRIVE's Mental Representation of the road environment, as perceived and understood by the model at this time (i.e. its current Situation Awareness), and the bottom right view illustrates the envelope-zone theory used in COSMODRIVE to support Decision Making and Action Planning, to support driving behaviours.

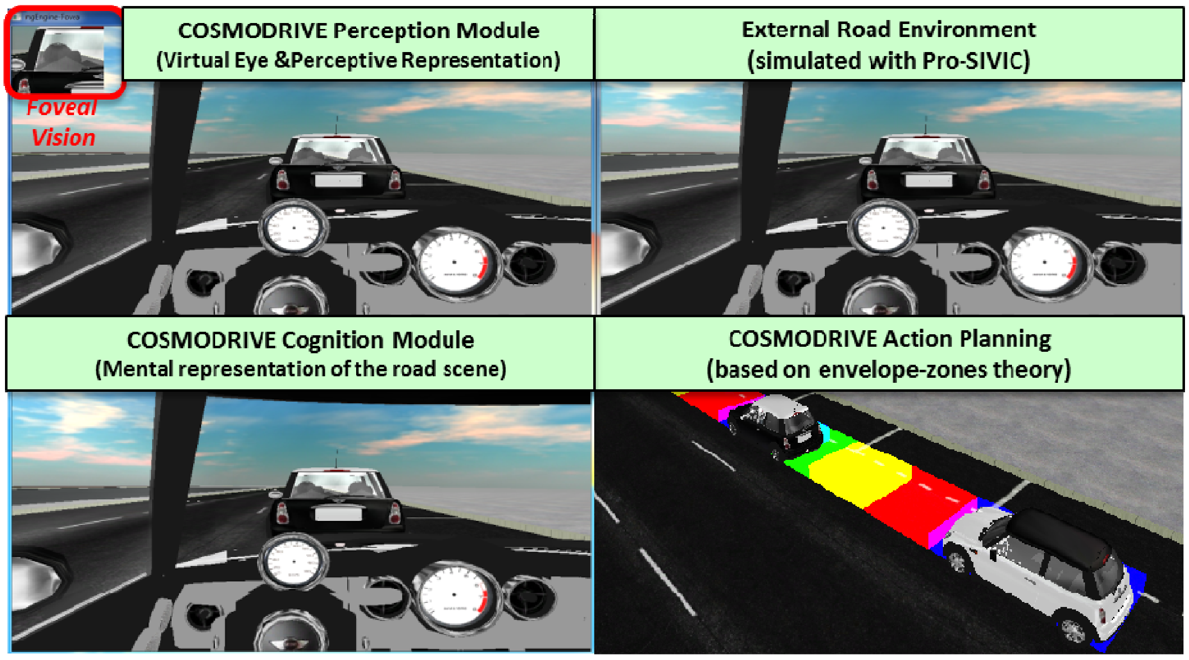

COSMODRIVE Perception Module
(Virtual Eye &Perceptive Representation)
External Road Environment
(simulated with Pro-SIVIC)
Foveal Vision
COSMODRIVE Cognition Module
(Mental representation of the road scene)
COSMODRIVE Action Planning
(based on envelope-zones theory)

Figure 49: COSMODRIVE's Representation and Virtual Eye

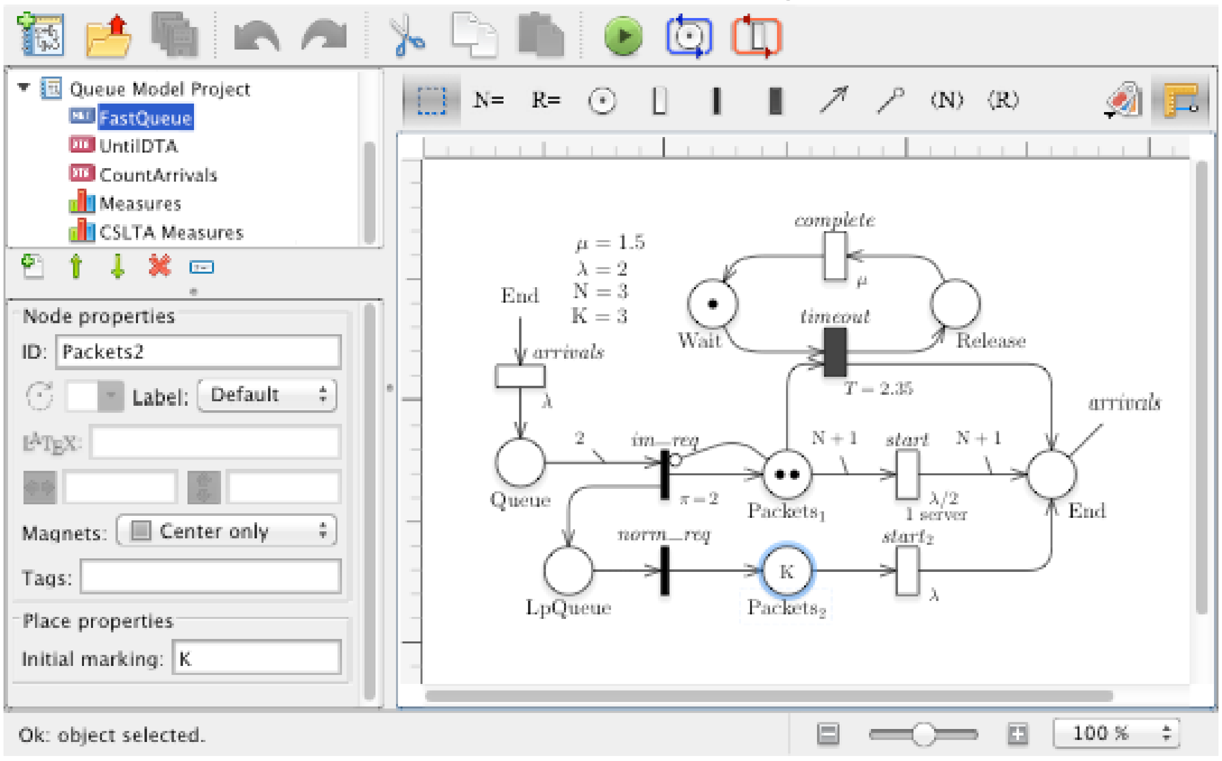
This COSMODRIVE/RTMaps/Pro-SiVIC tool chain is fully operational to progressively integrate pieces of information extracted in the road environment at the different levels of the human perceptive and cognitive system. Moreover, the RTMaps diagram shown in Figure 47 can easily be extended to allow other HoliDes MTTs, for interacting with both Pro-SiVIC and COSMODRIVE.

3.6 GreatSPN for MDPN (UTO)

3.6.1 Introduction

GreatSPN is a tool developed by the University of Torino in the last 30 years. 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 3.6.3

The GUI allows drawing the models graphically, using the Petri net formalism. The interface of the GUI is shown in Figure 50.

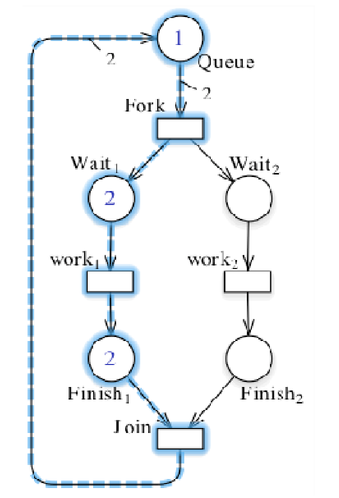

Queue Model Project
FastQueue
UntilDTA
CountArrivals
Measures
CSLTA Measures

Node properties
ID: Packets2
Label: Default
Magnets: Center only
Tags:
Place properties
Initial marking: K

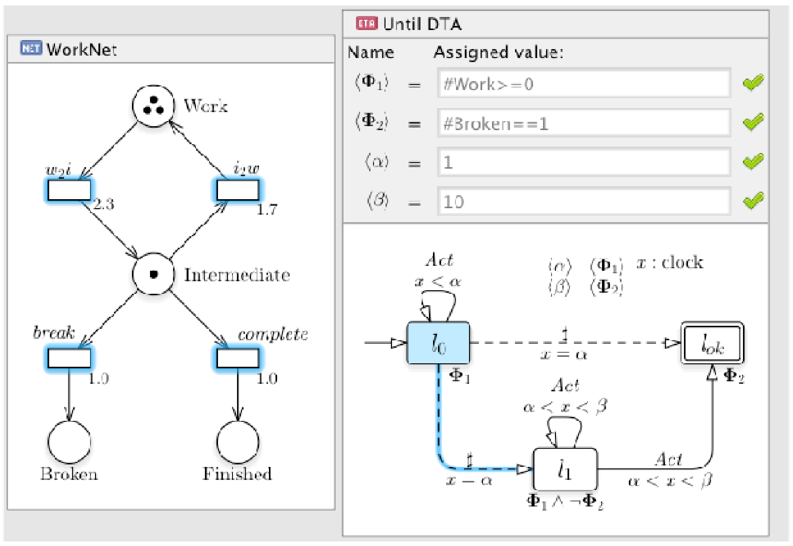
Ok: object selected.
100 %

Figure 50: 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 51.



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



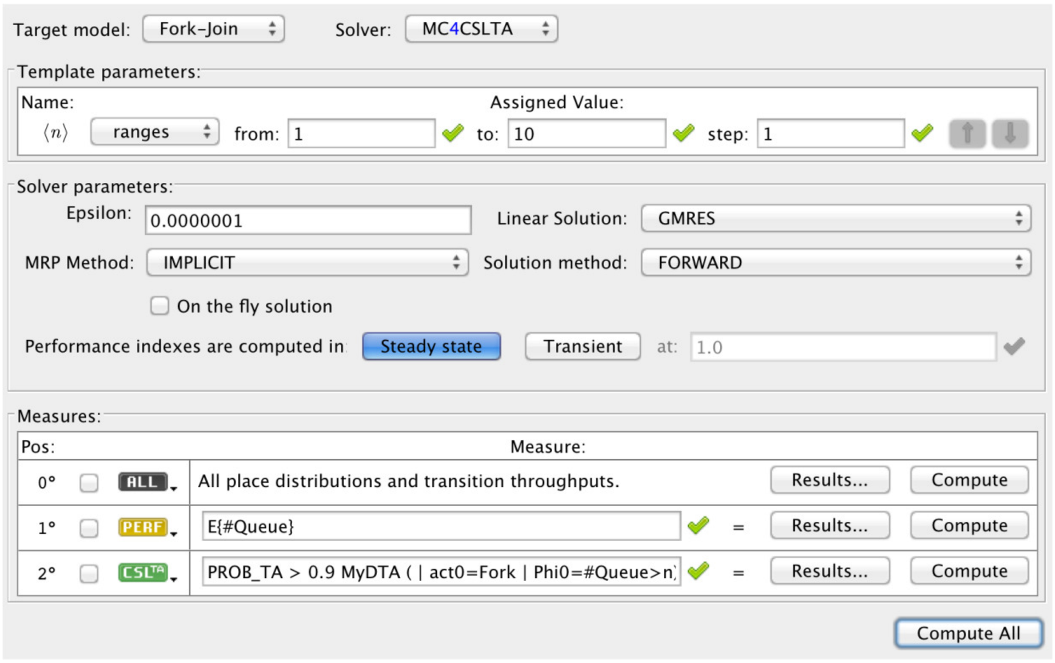
Name		Assigned value:
$\langle\Phi_1\rangle$	=	#Work>=0
$\langle\Phi_2\rangle$	=	#Broken==1
$\langle\alpha\rangle$	=	1
$\langle\beta\rangle$	=	10

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

Figure 51: 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 one 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 52 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:
⟨n⟩ ranges	from: 1 ✓ to: 10 ✓ step: 1 ✓

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°	ALL		All place distributions and transition throughputs.		Results...	Compute
1°	PERF		E{#Queue} ✓	=	Results...	Compute
2°	CSLTA		PROB_TA > 0.9 MyDTA (| act0=Fork | Phi0=#Queue>n) ✓	=	Results...	Compute

Compute All

Figure 52: 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 53, which displays an MDPN model drawn with the GreatSPN GUI.

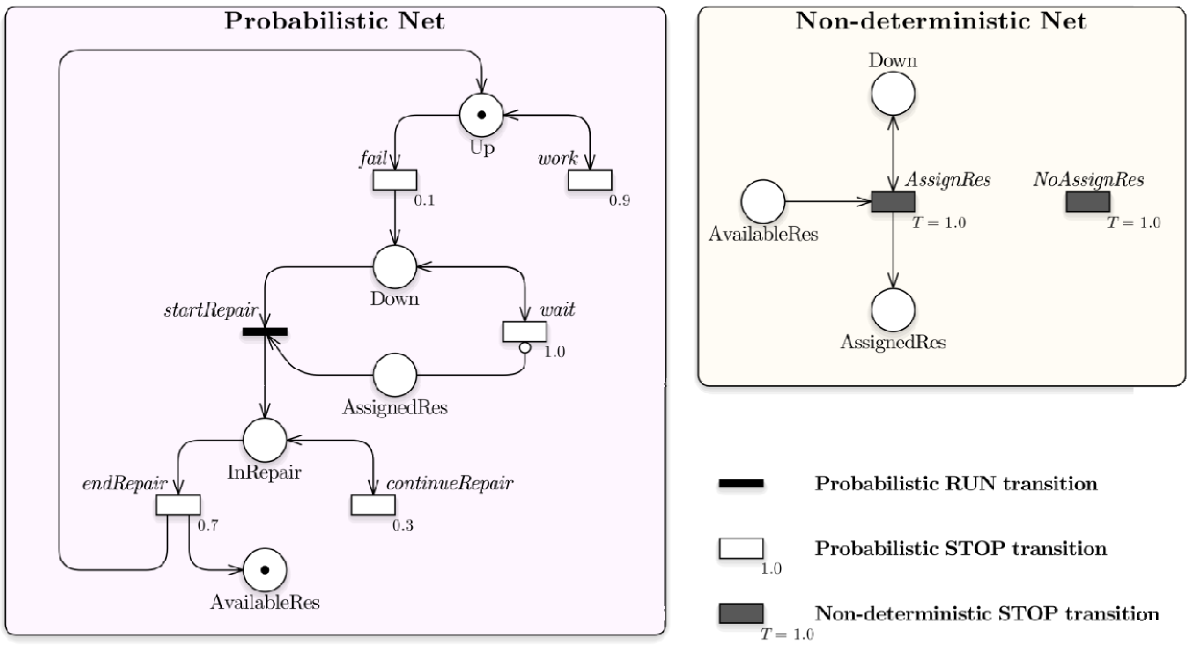


Figure 53: An MDPN in GreatSPN

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.

3.6.2 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 [53] 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 [19] 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 [3] 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 [45] for an overview of a number of these.

3.6.3 MTT Description

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

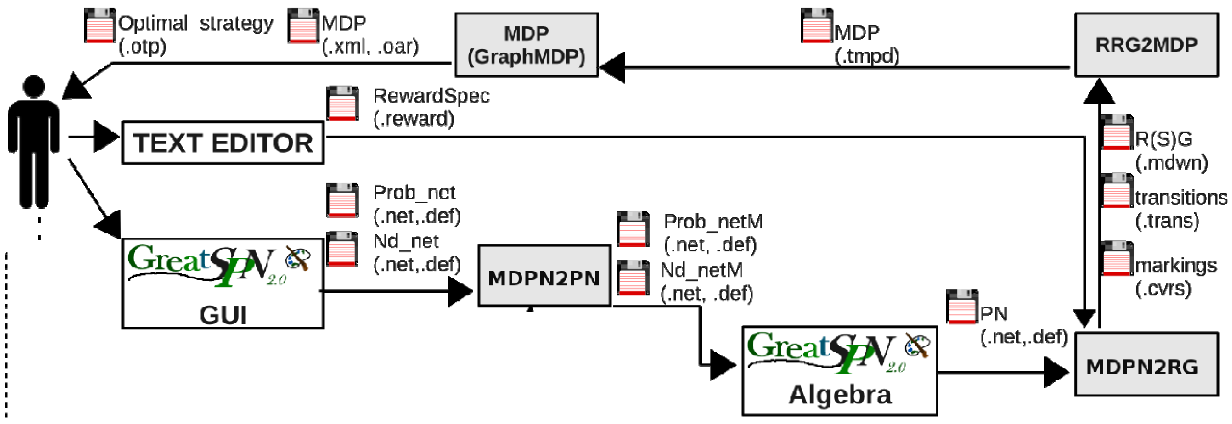


Figure 54: MDPN solver Architecture

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 54. The user must specify PN^{nd} and PN^{pr} subnets (in Figure 3 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.

3.6.4 RTP Integration Plan

As well discussed in Deliverable D4.4, the integration of MDP solvers for their use at run-time has needs that go beyond the use of the RTP platform based on the OSLC framework and that can be instead satisfied by the inclusion of the solvers into the RT-MAPS tool. The planned activity is to start with an integration of the model for the WP9 use case UC4 (lane change). The integration will require the construction of an RT-MAPS component of the MDPN solver that works in real time in the embedded controller of the car. The component should fulfil the following stream of actions:

- to read the continuous data streams from the car sensors (camera, lidars, accelerometers, …)
- to read the data streams that is produced by the analysis components (driver distraction and driver intention, as for example produced by the Bayesian model of the human operator)
- to generate (or to adapt a pre-generated) MDPN model whose parameters are computed using the data from the external sensors
- Compute the optimal strategy of the MDPN model, which provides input to the information warning intervention layer of the AdCoS architecture for the CRF test-vehicle.

It is at the time not yet decided whether the integration into RT-MAPS will include also the MDPN model solver (MdP generation and solution) or only the MdP solution, whilst leaving the MdP generation as an off-line task. Inside the Agents, Tasks and Resources (ATR) framework architecture of RT-MAPS, the Driver Model is an agent that uses the car sensors (resources) and interacts with the other agents that implement the cognitive model and the warning feedback.
Figure 55 shows the structure of the RT-MAPS component that contains the MDP solver of GreatSPN.

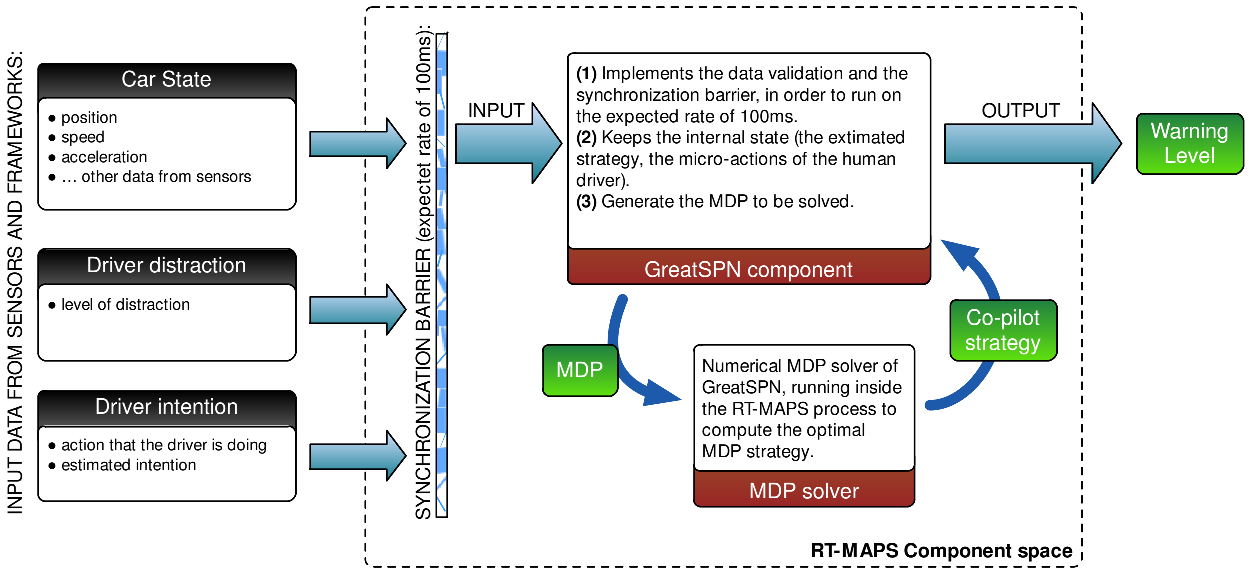

INPUT DATA FROM SENSORS AND FRAMEWORKS:
Car State
• position
• speed
• acceleration
• … other data from sensors
Driver distraction
• level of distraction
Driver intention
• action that the driver is doing
• estimated intention
SYNCHRONIZATION BARRIER (expectet rate of 100ms):
INPUT
(1) Implements the data validation and the synchronization barrier, in order to run on the expected rate of 100ms.
(2) Keeps the internal state (the extimated strategy, the micro-actions of the human driver).
(3) Generate the MDP to be solved.
GreatSPN component
OUTPUT
Warning Level
MDP
Numerical MDP solver of GreatSPN, running inside the RT-MAPS process to compute the optimal MDP strategy.
MDP solver
Co-pilot strategy
RT-MAPS Component space

Figure 55: Integration of MdP in RT-MAPS

The component expects in input a certain amount of data, provided by the sensors, and the intention/distraction classifiers, that are used to generate the solution MDP. However, since data from other component and sensors are not synchronized (each component/sensor work at its functional rate), a synchronization of these information is required. Synchronization can be done directly using the RT-MAPS synchronization facilities that are already implemented in the platform. At design stage, the estimated computation rate is of 100ms per cycle. Internally, the GreatSPN component keeps the last computed strategy as its state, and during each cycle it generates a new MDP with the updated input data, and re-computes the optimal strategy. The new strategy could be the same of the old strategy, or a new one. The output of the component is strictly related to the strategy itself, and is a synthetic warning level intended for the human driver.

3.7 Bayesian Autonomous Driver Mixture-of-Behaviours Models (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 (c.f., Section 2.1.4.2) 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.

3.7.1 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 56) 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, theses machine agents work without mutual interaction and adaptation. This can lead to unwanted warnings and interventions, which have the potential to annoy the driver to the point of disregarding or disabling the safety device, or even introduce new safety critical situations. 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.

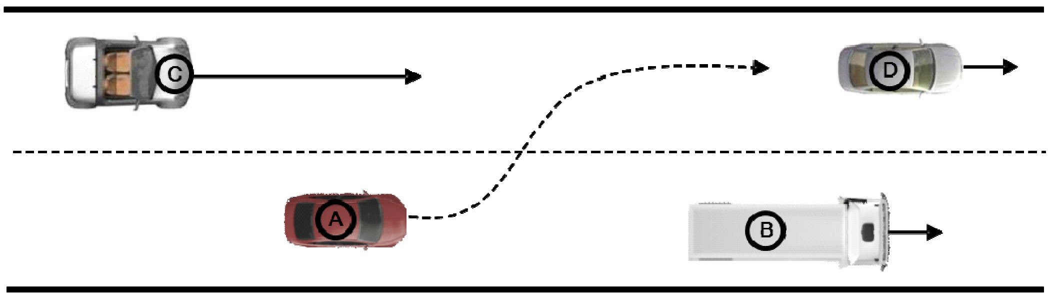


Figure 56: Representation of the target-scenario (the problem that the AdCoS intends to solve) in AUT domain.

Intention recognition is primarily concerned with the recognition of behaviour intentions, which are defined as "*a person's intentions to perform various behaviours*" [28]. 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., [60]).

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*" [2]. Following these implications, an intention can be seen as an immediate antecedent or predictor of the human's behaviour in the nearest future [2]. 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 [50]. 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 [2]. Accordingly, they have to be inferred from the available context.

3.7.2 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 [4]. 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 [84][85][64][102] and control-theoretic models for the autonomous stage [47][96][100][101]. These kinds of models are quite standard approaches now [22]. More recently, approaches for the autonomous and associative stage have been broadened by probabilistic driver models (e.g., [29][52][71][103][104]).

Driver models have been used in traffic scenario simulations to provide safety assertions and support risk-based design [22][100]. 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 [99], 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 [8][49][66][72][76], and in the last years, several studies on recognition of driver's intentions have been reported [66], addressing the recognition of lane-change intentions [17][26][69][83], braking actions [68][67], turning manoeuvres [59][26], and overall trajectory prediction [97].

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) [71][17][69][1][49][48][10][76][86][8]. 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 [72][76][26]). An additional overview of the state of the art for intention recognition can be found e.g., in [50] and [20].

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.

3.7.3 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.

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

Not all of these behavioural schemata 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 $\mathrm{Val}(I) = \{\text{lane change left, lane change right, default}\}$.

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 $\boldsymbol{A} = \{A_1, \dots, A_n\}$:

Variable	Type	Description
Steering angle	Continuous	Steering angle value
Acceleration	Continuous	Acceleration of the driver's vehicle
Head position	Continuous	Position of the driver's head
Head position rate of change	Continuous	Rate of change of the driver's head position
Head orientation	Continuous	Orientation of the driver's head (yaw, roll, pitch)
Head orientation rate of change	Continuous	Rate of change of the driver's head orientation
Direction Indicator signal	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 $\boldsymbol{O} = \{O_1, \dots, O_m\}$. 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
Lane curvature	Continuous	Curvature of the road
Lateral derivation	Continuous	Lateral distance between the middle of the lane and the longitudinal axis of the driver's vehicle
Yaw angle	Continuous	Angle between the longitudinal axis of the driver's vehicle and lane direction, tangent to the lane (also called "lane yaw angle")
Yaw angle rate of change	Continuous	Rate of change of the yaw angle
Lead car lat. speed	Continuous	Lateral velocity of the lead vehicle
Lead car lat. acceleration	Continuous	Lateral acceleration of the lead vehicle
Lead car long. speed	Continuous	Longitudinal velocity of the lead vehicle
Lead car long. acceleration	Continuous	Longitudinal acceleration of the lead vehicle
Lead car lat. distance	Continuous	Lateral distance of the lead vehicle in respect to the driver's vehicle
Lead car long. distance	Continuous	Longitudinal distance of the lead vehicle in respect to the driver's vehicle
THW to lead car	Continuous	Time headway to the lead vehicle
THW to lead car change of rate	Continuous	Rate of change of the time headway to the lead vehicle
TTC to lead car	Continuous	Time-to-contact to the lead vehicle
TTC to lead car rate of change	Continuous	Rate of change of the time-to-contact to the lead vehicle
Velocity difference	Continuous	Difference between the velocities of the driver's and the lead vehicle
Velocity difference rate	Continuous	Rate of change of the difference between the velocities of the driver's and the lead vehicle
Velocity	Continuous	Velocity of the driver's vehicle
VDD	Discrete	Visual Distraction Detection
VTSD	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.

3.7.3.2 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 57. 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).

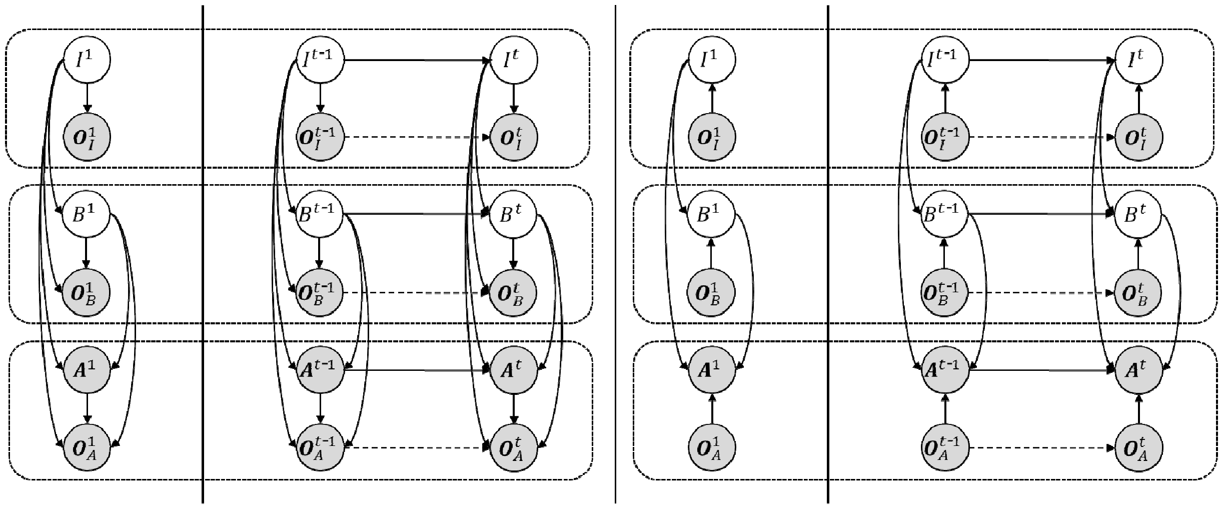


Figure 57: Template structures of BAD MoB models for intention recognition, (loosely) based on FHMMs (left) and HMDTs (right), both defined by a Bayesian network for the first time-slice $t = 1$ and a 2TBN for all $t > 1$. Blank nodes represent

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.

3.7.3.2.1 Generative Approach

For the generative approach we need to define a factorization of the JPD $P(I^{1:T}, B^{1:T}, \boldsymbol{A}^{1:T}, \boldsymbol{O}^{1:T})$. The chain rule of probabilities [51] allows without any independency assumptions to factorize the JPD as:

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

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:

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

The assumption of stationary processes then allows us to use a single 2TBN $P(I^t, B^t, \boldsymbol{A}^t, \boldsymbol{O}^t | I^{t-1}, B^{t-1}, \boldsymbol{A}^{t-1}, \boldsymbol{O}^{t-1})$ 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 $P(I^t, B^t, \boldsymbol{A}^t, \boldsymbol{O}^t | I^{t-1}, B^{t-1}, \boldsymbol{A}^{t-1}, \boldsymbol{O}^{t-1})$ as:

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

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

- $P(I^t|I^{t-1},B^{t-1},\boldsymbol{A}^{t-1},\boldsymbol{O}^{t-1}) = P(I^t|I^{t-1})$: 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 $\boldsymbol{A}^{t-1}$, and observations $\boldsymbol{O}^{t-1}$. 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.
- $P(B^t|I^{t-1:t},B^{t-1},\boldsymbol{A}^{t-1},\boldsymbol{O}^{t-1}) = P(B^t|I^t,B^{t-1})$: 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 $\boldsymbol{A}^{t-1}$, and observations $\boldsymbol{O}^{t-1}$. 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 gain additional knowledge from $\boldsymbol{A}^{t-1}$. Once again, the main controversy lies in the conditional independence from $\boldsymbol{O}^{t-1}$.
- $P(\boldsymbol{A}^t|I^{t-1:t},B^{t-1:t},\boldsymbol{A}^{t-1},\boldsymbol{O}^{t-1}) = P(\boldsymbol{A}^t|I^t,B^t,\boldsymbol{A}^{t-1})$: Given the current intention I^t, the current manoeuvre B^t and the previous actions $\boldsymbol{A}^{t-1}$, the current actions $\boldsymbol{A}^t$ are conditionally independent of the previous intention I^{t-1}, the previous behaviour B^{t-1}, and the previous observations $\boldsymbol{O}^{t-1}$. 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.
- $P(\boldsymbol{O}^t|I^{t-1:t},B^{t-1:t},\boldsymbol{A}^{t-1:t},\boldsymbol{O}^{t-1}) = P(\boldsymbol{O}^t|I^t,B^t,\boldsymbol{A}^t,(\boldsymbol{O}^{t-1}))$: Given the current intention I^t, behaviour B^t, and actions $\boldsymbol{A}^t$ (and potentially the previous observations $\boldsymbol{O}^{t-1}$), the current observations $\boldsymbol{O}^t$ are conditionally independent from the previous intention I^{t-1}, behaviour B^{t-1}, and actions $\boldsymbol{A}^{t-1}$. 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 $P(\boldsymbol{O}^t|I^t,B^t,\boldsymbol{A}^t,(\boldsymbol{O}^{t-1}))$ as $P(\boldsymbol{O}_I^t|I^t(\boldsymbol{O}_I^{t-1}))P(\boldsymbol{O}_B^t|I^t,B^t,(\boldsymbol{O}_B^{t-1}))P(\boldsymbol{O}_A^t|I^t,B^t,\boldsymbol{A}^t,(\boldsymbol{O}_A^{t-1}))$, i.e., independent sets of observations relevant for intentions, behaviours, and actions.

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

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

The graph structure of this template is shown in Figure 57 (left), which can be seen as a modification of a model class known as Factorial Hidden Markov Models (FHMM) [32].

Concerning the finer factorization of $P(\boldsymbol{A}^t|\boldsymbol{A}^{t-1},B^t,I^t)$, 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 $P(\boldsymbol{A}^t|\boldsymbol{A}^{t-1},B^t,I^t) = \prod_{i=1}^{n} P(A_i^t|A_i^{t-1},B^t,I^t)$. Unfortunately, in general, we can’t make the same assumptions about the environment, i.e., $P(\boldsymbol{O}^t|(\boldsymbol{O}^{t-1}),\boldsymbol{A}^t,B^t,I^t) \neq \prod_{j=1}^{m} P\left(O_j^t|(O_j^{t-1}),\boldsymbol{A}^t,B^t,I^t\right)$. 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 $P(\boldsymbol{O}^t|(\boldsymbol{O}^{t-1}),\boldsymbol{A}^t,B^t,I^t)$ that will primarily be guided by the need for computational efficiency and robust parameter estimation.

3.7.3.2.2 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 $P(I^{1:T},B^{1:T},\boldsymbol{A}^{1:T}|\boldsymbol{o}^{1:T})$. Applying the same independency assumptions as discussed for the generative approach, we obtain the following factorization:

$$P(I^{1:T}, B^{1:T}, \boldsymbol{A}^{1:T} | \boldsymbol{o}^{1:T}) \\ = P(I^1 | \boldsymbol{o}_I^1) P(B^1 | I^1, \boldsymbol{o}_B^1) P(\boldsymbol{A}^1 | B^1, I^1, \boldsymbol{o}_A^1) \\ \prod_{t=2}^{T} P(I^t | I^{t-1}, \boldsymbol{o}_I^t) P(B^t | B^{t-1}, I^t, \boldsymbol{o}_B^t) P(\boldsymbol{A}^t | \boldsymbol{A}^{t-1}, B^t, I^t, \boldsymbol{o}_A^t).$$

The graph structure of this template is shown in Figure 57 (right), which can be seen as a modification of a model class known as Hidden Markov Decision Trees (HMDT) [46].

Apparently, the biggest advantage of the above factorization lies in the fact that we don't need to distinctively model the observation sequence $\boldsymbol{o}^{1:T}$, 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 $\boldsymbol{o}^{t-1}$ given the current observations $\boldsymbol{o}^t$. Note that although we assume that actions are always observable, we will explicitly include them in our discriminative model $P(I^{1:T}, B^{1:T}, \boldsymbol{A}^{1:T} | \boldsymbol{o}^{1:T})$. 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 $P(\boldsymbol{A}^{j:t} | \boldsymbol{a}^{1:j-1}, \boldsymbol{o}^{1:t})$. 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 $P(\boldsymbol{A}^t | \boldsymbol{A}^{t-1}, B^t, I^t, \boldsymbol{o}_A^t)$, we will once again start with the assumption of independent Markov chains, i.e., $P(\boldsymbol{A}^t | \boldsymbol{A}^{t-1}, B^t, I^t, \boldsymbol{o}_A^t) = \prod_{i=1}^{m} P(A_i^t | A_i^{t-1}, B^t, I^t, \boldsymbol{o}_A^t)$, where we will try to model each CPD $P(A_i^t | A_i^{t-1}, B^t, I^t, \boldsymbol{o}_A^t)$ by a conditional sub-network $\frac{1}{Z} P(A_i^t, \boldsymbol{o}_A^t | A_i^{t-1}, B^t, I^t)$, where Z denotes a normalization factor $Z = \sum_{a_i \in A_i} P(a_i^t, \boldsymbol{o}_A^t | A_i^{t-1}, B^t, I^t)$, in the case of a discrete variable A_i, or resp. $Z = \int_{-\infty}^{\infty} P\left(a_j^t, \boldsymbol{o}_A^t | A_j^{t-1}, B^t, I^t\right) da_j^t$ 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 [90].

3.7.3.3 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: $P(I^t, B^t | \boldsymbol{a}^{0:t}, \boldsymbol{o}^{0:t})$. Given this joint belief state, we can easily derive the marginal belief states of intentions $P(I^t | \boldsymbol{a}^{0:t}, \boldsymbol{o}^{0:t})$ and behaviours $P(B^t | \boldsymbol{a}^{0:t}, \boldsymbol{o}^{0:t})$. The estimation of the belief state is known as filtering and can, for both template structures, be solved in *constant* time by recursively computing $P(I^t, B^t | \boldsymbol{a}^{0:t}, \boldsymbol{o}^{0:t})$ from the previous belief state $P(I^{t-1}, B^{t-1} | \boldsymbol{a}^{0:t-1}, \boldsymbol{o}^{0:t-1})$.

3.7.4 RTP Integration Plan

A BAD MoB model is an instantiation of a DBN that complies to the modelling language defined in Section 2.1.4.2.3. As such, it can’t be seen as a standalone tool that will be integrated in the HF-RTP. However, OFFIS will implement an OSLC compliant web-service, which allows to retrieve all available BAD MoB models from a database, planned to release by the 30.06.2015.

Within HoliDes, available BAD MoB models will be utilized by the Driver Intention Recognition (DIR) module. The DIR module will be implemented in RTMaps, which is part of the HoliDes Tooling landscape that composes the HF-RTP (see D1.3). The main objective of the DIR module is the assessment of the current intentions and behaviours of a single human agent, i.e., the driver of vehicle “A” depicted in Figure 56, within the use-cases for adapted assistance (see D9.3).

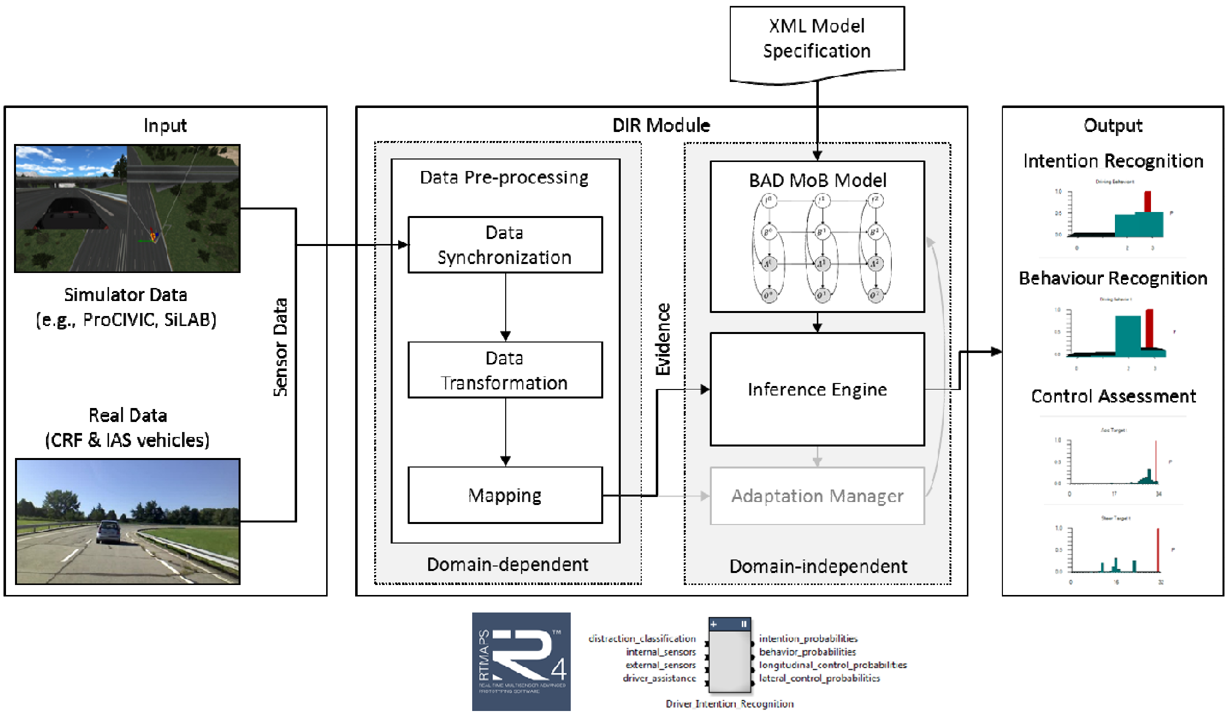


Figure 58: Overview of the DIR module, to be provided as in RTMaps.

As depicted in Figure 58, the DIR module consists of two parts, a domain- and task-*dependent* part, tailored to the actual system architecture and specification of the AdCoS for adapted assistance that deals with pre-processing the available input, and a domain- and task-*independent* part consisting of an *inference engine* that enables the DIR module to answer probabilistic queries according to an arbitrary probabilistic model defined in an xml-specification. Both parts are implemented as separate RTMaps components, so that the domain- and task-independent inference engine can potentially be used in different domains utilizing different probabilistic models. Within RTMaps, the inference engine provides an interface to select an xml model-specification. In general, any specification that satisfies the modelling language described in Section 2.1.4.2.3 can be utilized. For the DIR module, the inference engine will be used to constantly, at each time step t, infer the joint belief state of intentions and behaviours given the all available evidence about actions and context observations observed up to this point: $P(I^t, B^t | \boldsymbol{a}^{0:t}, \boldsymbol{o}^{0:t})$. Given this joint belief state, the DIR module can provide the marginal belief states for intentions $P(I^t | \boldsymbol{a}^{0:t}, \boldsymbol{o}^{0:t})$ and behaviours $P(B^t | \boldsymbol{a}^{0:t}, \boldsymbol{o}^{0:t})$. Figure 59 shows the embedding of the DIR module in the overall AdCoS for adapted assistance. A more detailed description of the DIR module and the Automotive AdCoS for adapted assistance can be found in D9.3.

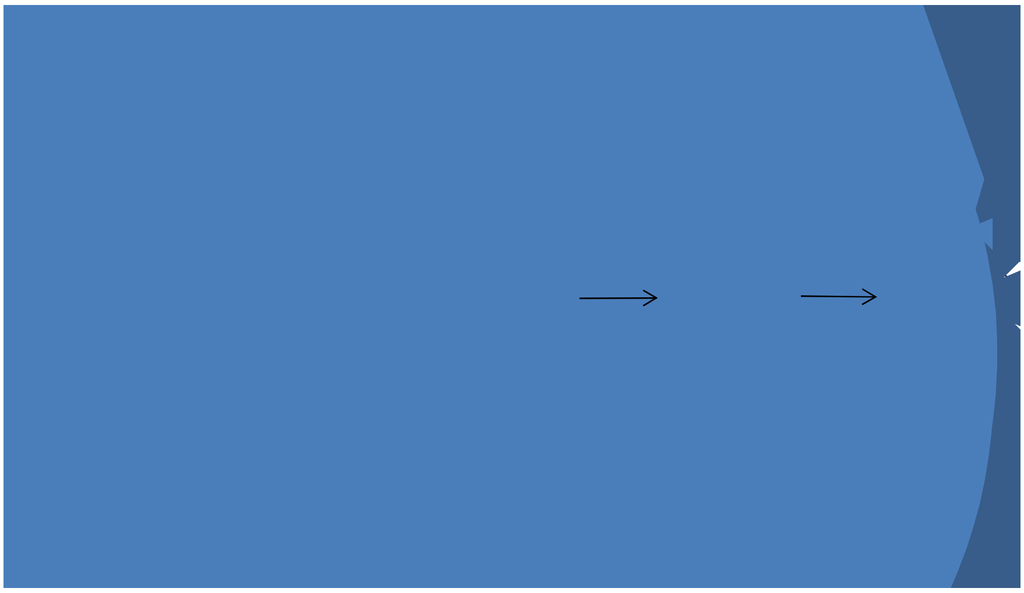


Figure 59: Overview of the architectural scheme of the AdCoS application for adaptive assistance in WP9.

3.8 Detection of driver distraction based on in-car measures (TWT)

3.8.1 Introduction

Distraction during driving leads to a delay in recognition of information that is necessary to safely perform the driving task [81]. Thus, distraction is one of the most frequent causes for car accidents [9], [37]. 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 [91]. 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 of 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.

3.8.2 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 [56], [58]. The recent dynamic Bayesian model by Liang and Lee [57] 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.

3.8.3 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. A detailed description of those systems can be found in D9.2.
In addition, deriving knowledge about the human operator can be very valuable in the system validation phase. While interacting with a prototype or some modules of the AdCoS, the operator's degree of distraction can be evaluated. 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 and face-tracking data from the driver. These data need to be stored in a way that enables linking them to certain system states, e.g., inputs from the user to the system. Thus multimodal data integration and synchronization needs to be guaranteed.
The tool provides a temporal description of the driver's degree of distraction. We will thus use a continuous measure provided by a regression analysis. 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 INTEMPORA. Personal components of the cognitive model and computations are intended to be used as a user profile that potentially can be used by other systems with the same model. This user profile could, for instance, be transferred to other cars.
For integration of the tool into the HF-RTP, its usage during system validation phase plays an essential role and focusses on its output parameters which, in this case, still need to be detailed.

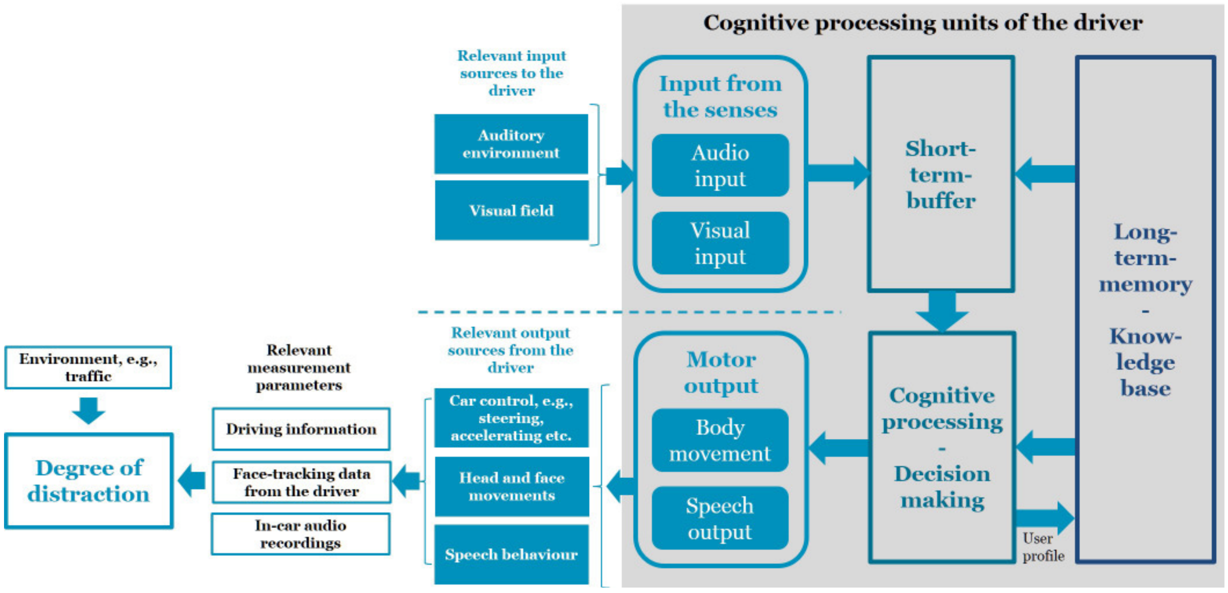

Cognitive processing units of the driver
Relevant input sources to the driver
Auditory environment
Visual field
Input from the senses
Audio input
Visual input
Short-term-buffer
Long-term-memory - Know-ledge base
Environment, e.g., traffic
Degree of distraction
Relevant measurement parameters
Driving information
Face-tracking data from the driver
In-car audio recordings
Relevant output sources from the driver
Car control, e.g., steering, accelerating etc.
Head and face movements
Speech behaviour
Motor output
Body movement
Speech output
Cognitive processing - Decision making
User profile

Figure 60. Architectural design of the cognitive model predicting the driver's distraction degree.

Figure 60 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 parameters derived from in-car audio recordings, face-tracking information, car information (e.g. driving parameters) and environmental information like the distance to the pace car to be followed.

3.8.4 RTP Integration Plan

In the system validation phase, deriving knowledge about the human operator can be very valuable. While interacting with a prototype or some modules of the AdCoS, the operator's degree of distraction can be evaluated. The tool provides feedback whether or not a new system (module) increases or decreases the operator's degree of distraction. The output of the MTT addresses in this case the system developer and thus must be part of the development workflow. Here, the multi-modal nature of the distraction estimation plays an essential role since it may provide the system developer with more details about the cause of the distraction.

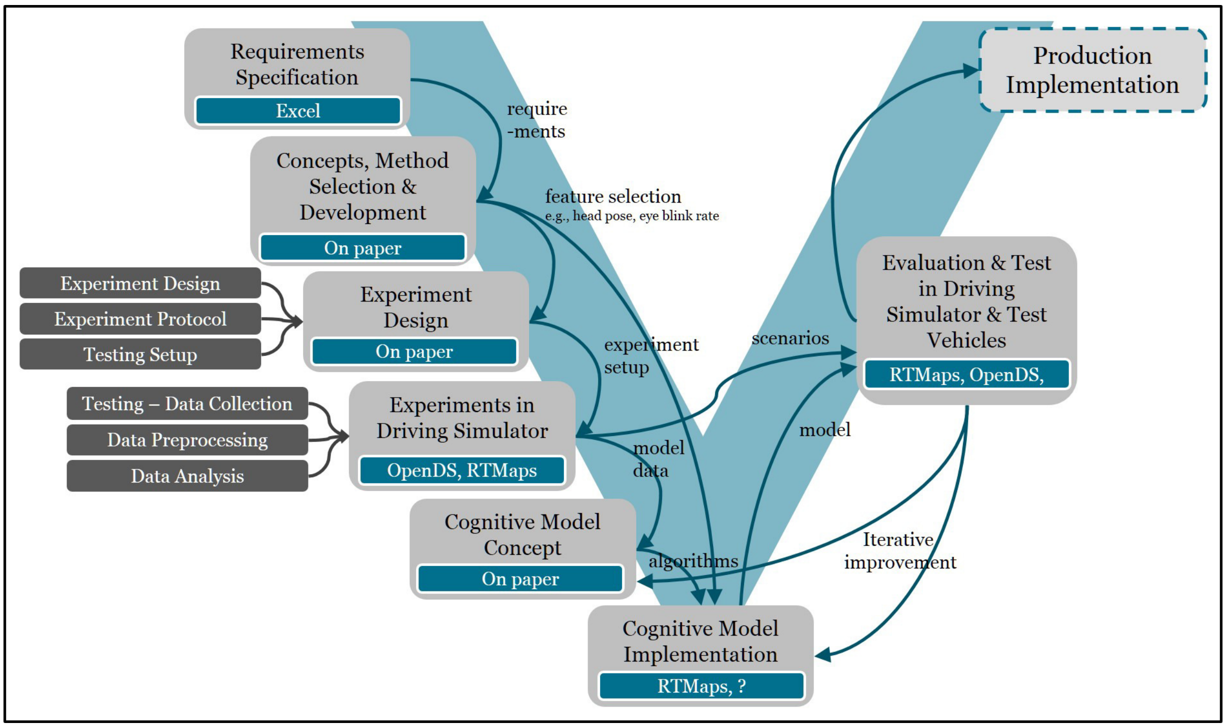


Figure 61. Workflow with potential tools to be used for the development of the AdCoS containing the driver distraction estimation.

Figure 61 shows the individual steps of the workflow integrating the distraction estimation 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. On the basis of this data the cognitive model will be implemented and evaluated and validated using simulator experiments.
Some steps still lack proper tool-support. With the HF-RTP, we expect further refinements of this workflow regarding potential tools to cover these steps.

3.9 Djnn (ENA)

The Djnn tool is described in D4.4 in more detail. In order to not duplicate input, please refer to D4.4 for information on Djnn.

4 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. The AdCoS requirements have been analysed, because it may be the case that also an AdCoS requirement is relevant for a WP model.

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, applied in WP5
FaceLab 5 + Eyeworks (SNV)	3rd Party tool not developed by partner, applied in WP5
Captiv T-sens (SNV)	3rd Party tool not developed by partner, applied in WP5
Enobio (SNV)	3rd Party tool not developed by partner, applied in WP5
HS-Searchopt	Replaced by other tool in WP7, details will be added in next deliverable version

The following table gives an overview on the current status:

Status	**Total**	**not relevant**	**need feedback**	**assigned**	**accepted**	**rejected**	**in progress**	**in test**	**fulfilled**
RTP req.:	176	89	8	31	12	9	25	12	4
AdCoS req.:	264	215	3	41	1	5	3	1	0
Total	**440**	**304**	**11**	**72**	**13**	**14**	**28**	**13**	**4**

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.

5 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:

- Initial HMI Interaction, Training, Task- and Resource Modelling languages have been formalized
- Formalizations of the MDP/MDPN and the DBN languages have been started as behavioural operator models; cognitive models will follow
- 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.

There is a clear progress in the requirements visible, in the initial requirements analysis, no requirements had the status “in progress, in test or fulfilled”. 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.

6 References

[1] 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.

[2] 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.

[3] 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.

[4] Anderson, J.R. (2000). Learning and Memory: An Integrated Approach. 2nd Edition, Wiley.

[5] Anderson, J.R. and Lebiere, C.J.: The Atomic Components of Thought. Lawrence Erlbaum Associates, June 1998.

[6] 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.

[7] Annett, J.: Hierarchical Task Analysis. In: E. Hollnagel (ed). Handbook of Cognitive Task Design. Lawrence Erlbaum Associates, Mahwah, New Jersey, 17-35, 2003

[8] 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.

[9] 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

[10] Bando, T., Miyahara, T., Tamatsu, Y. (2011): Traffic Interactions: Estimate Driving Behavior's Influence. In 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 546-551, Baden-Baden, Germany, June 5-9, 2011, IEEE Catalog Number: CFP11IVS-CDR, ISBN: 978-1-4577-0889-3.

[11] 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

[12] 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

[13] 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

[14] 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

[15] 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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] Börger, J. (2013). Fahrerintentionserkennung und Kursprädiktion mit erweiterten Maschinellen Lernverfahren. Dissertation, Universität Ulm.

[21] Bruseberg, A. & Shepherd, A.: Job Design in Integrated Mail Processing. In: D. Harris (ed) Engineering Psychology and Cognitive Ergonomics. Volume Two: Job Design and Product Design. Ashgate Publishing, Aldershot, Hampshire, (25-32), 1997.

[22] Cacciabue, P.C. (ed.) (2007). Modelling Driver Behaviour in Automotive Environments. London: Springer, ISBN-10: 1-84628-617-4.

[23] Card, S. K., Moran, T. P. & Newell, A.: The Psychology of Human-Computer Interaction. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers, 1983.

[24] Cooper, R. and Fox, J.: Cogent: A visual design environment for cognitive modeling. Behavior Research Methods, Instruments, & Computers, 30(4):553–564, 1998

[25] 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.

[26] 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.

[27] Dzaack-J., Wiesner A., Urbas L. & Heinath, M.: Simplifying the development and the analysis of cognitive models. In Proceedings of EuroCogSci07, Delphi, Greece, 2007.

[28] Fishbein, M., Ajzen, I. (1975). Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research. Reading, MA: Addison-Wesley.

[29] 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.

[30] Foyle, C.F. & Hooey, B.L.: Human Performance Modeling in Aviation. CRC Press, Taylor & Francis Group, Boca Raton, FL, 2008.

[31] Freed, M.A.: Simulating Human Performance in Complex, Dynamic Environments. Dissertation, Northwestern University, Evanston, Illinois, June 1998

[32] Ghahramani, Z., Jordan, M. I. (1997). Factorial Hidden Markov Models. In Machine Learning, p.1-31

[33] 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.
[34] Hartson, H. R. and Gray, P. D.: Temporal aspects of tasks in the user action notation. Human Computer Interaction 7, pp. 1-45, 1992
[35] 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
[36] Hollnagel, E.: Human Reliability Analysis: Context and Control. London: Academic Press London, 1993.
[37] 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.
[38] 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.
[39] Itti, L. and Koch, C. Computational Modeling of Visual Attention, Nature Reviews Neuroscience 2(3):194-203 2001
[40] 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
[41] 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.
[42] Jenkins, D. P., Stanton, N.A., Walker, G.H., Salmon, P.M.: Cognitive Work Analysis: Coping with Complexity, 2012, Ashgate Publishing.
[43] 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.
[44] 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

[45] Jonsson, B., Larsen, K. G., and Yi, W.: Probabilistic extensions of process algebras. In Handbook of Process Algebra, pages 685–710. Elsevier, 2001.

[46] 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.

[47] 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.

[48] 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.

[49] Kasper, D., Weidl, G., Dang, T., Breuel, G., Tanke, A., Rosenstiel, W. (2011). Object-Oriented Bayesian Networks for Detection of Lane Change Meneuvers, 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.

[50] Kobiela, F. (2011). Fahrerintentionserkennung für autonome Notbremssysteme. Springer.

[51] Koller, D., Friedman, N. (2009). Probabilistic Graphical Models: Principles and Techniques. MIT Press.

[52] Kumugai, T., Akamatsu, M. (2006). Prediction of Human Driving Behavior Using Dynamic Bayesian Networks. In IEEE Trans. Inf. & Syst., Vol. E89-D, No.2.

[53] 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.

[54] 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

[55] 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
[56] 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
[57] Liang, Y., Lee, J. D.: A hybrid bayesian network apporach to detect driver cognitive distraction. Transportation Research Part C 38, pp. 146-155, 2014
[58] 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
[59] 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.
[60] Liu. A., Pentland, A. (1997). Towards real-time recognition of driver intentions. In Intelligent Transportation System, 1997. ITSC’97, pp. 236-241.
[61] Lüdtke, 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.
[62] Lüdtke, 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
[63] Lüdtke, 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.
[64] Lüdtke, 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
[65] Lüdtke, 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

[66] 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.

[67] McCall, J. C., Trivedi, M. (2007). Driver Behavior and Situation Aware Brake Assistance for Intelligent Vehicles. In Proeedings. of the IEEE, vol. 95, no. 2, pp. 374-386.

[68] McCall, J. C., Trivendi, M. (2006). Human behaviour based predictive brake assistance. In Proceedings of the 2006 IEEE Intelligent Vehicles Symposium, pp. 8-12.

[69] 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.

[70] 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

[71] 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.

[72] 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.

[73] Neisser, U.: Cognition and reality: principles and implications of cognitive psychology. W.H.Freeman, San Francisco, 1976

[74] Newell, A., & Simon, H. A.: GPS, a program that simulates human thought. In Heinz Billing, Hrsg., Lernende Automaten, S. 109-124. Oldenbourg, München, 1961
[75] Ormerod, T. C. and Shepherd, A.: Using task analysis for information requirements specification: The SGT method. Lawrence Erlbaum Associates, 2004
[76] 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.
[77] Osterloh, J.-P., Bracker, H., Müller, H., Kelsch, J., Schneider, B., & Lüdtke, 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
[78] Paternò, F.: Model-Based Design and Evaluation of Interactive Applications. Applied Computing. Springer-Verlag, Berlin. ISBN: 185233155, 1999
[79] Piso, E.: Task analysis for process-control tasks: The method of Annett et al. applied. Occupational Psychology 54, 347-254, 1981
[80] Puterman, M.L.: Markov Decision Processes. Discrete Stochastic Dynamic Programming, Wiley, Chichester, 2005
[81] 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
[82] 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.
[83] 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.
[84] 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.
[85] Salvucci, D. D., Gray, R. (2004). A two-point visual control model of steering. In Perception, 33, pp.1233-1248.

[86] 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.
[87] 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
[88] 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
[89] Stanton, N. A.: Hierarchical task analysis: developments, applications and extensions. Applied Ergonomics, 37, (1), 55-79, 2006
[90] Su, J., Zhang, H., Ling, Ch., L., Matwin, S. (2008). Discriminative Parameter Learning for Bayesian Networks. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland.
[91] 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
[92] 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.
[93] Vorndran, I.. „Unfallentwicklung auf deutschen Straßen 2010", Statistisches Bundesamt Deutschland - Destatis, 2010
[94] W3C: MBUI - Task Models, W3C Working Group Note 08 April 2014 (http://www.w3.org/TR/task-models/), 2014
[95] Weber, L., Steenken, R., Lüdtke, A.: „Integrated Modeling for Safe Transportation (IMoST2) - Driver Modeling & Simulation", 4. Berliner Fachtagung Fahrermodellierung. June 13th - 14th. Berlin 2013.
[96] 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.
[97] 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.

[98] Wortelen, B., Lüdtke, 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

[99] Yangsheng, X., Ka Keung, C. L. (2005). Human Behavior Learning and Transfer, CRC Press Inc.

[100] Brackstone, M., McDonald, M. (1999). Car-following: a historical review. In Transportation Research Part F, 2, pp. 181-196.

[101] Ahmed, K., I. (1999). Modeling Driver's Acceleration and Lane Changing Behavior. PhD thesis, Massachusetts Institute of Technology.

[102] Aasman, J. (1995). Modeling driver behaviour in SOAR. In Leidschendam, The Netherlands: KPN Research.

[103] 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.

[104] 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 15th World Congress on Intelligent Transport Systems and ITS America's 2008 Annual Meeting.

[105] Jean-Michel Hoc. Human and automation: a matter of cooperation. Pruski, A. HUMAN 07, 2007, Timimoun, Algeria. Université de Metz, pp.277-285.