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

D 2.7 – Modelling Techniques and Tools Vs2.0

Project Number:	332933
Classification:	Confidential
Work Package(s):	WP2
Milestone:	M5
Document Version:	V0.1
Issue Date:	28.06.2016
Document Timescale:	Project Start Date: October 1, 2013
Start of the Document:	30
Final version due:	36
Deliverable Overview:	**Main document:** Modelling Techniques and Tools V2.0, **Annex I:** Requirements (confidential), **Annex II:** SyllabusManager Manual (confidential)
Compiled by:	Sonja Cornelsen - TWT
Authors:	Sonja Cornelsen, Iris Cohen, Marieke Thurlings – TWT Jan-Patrick Osterloh, Sebastian Feuerstack, Mark Eilers – OFF Roberta Presta – SNV Thierry Bellet, Jean-Charles Bornard, Bertrand Richard – IFS Mathieu Magnaudet – ENA Sara Sillaurren Landaburu – TEC
Reviewers:	Daniel Prun – ENAC, Tango Fabio – CRF
Technical Approval:	Jens Gärtner, Airbus Group Innovations
Issue Authorisation:	Sebastian Feuerstack, OFF

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

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

RECORD OF REVISION		
Date (DD.MM.YY)	Status Description	Author
05.07.2016	Initial Structure	Sonja Cornelsen (TWT)
12.08.2016	TEC contribution	Jesus Lopez Lobo
12.08.2016	TWT contribution	Cornelsen/Cohen
19.08.2016	OFF contribution	JPO, SF, ME
12.08.2016	SVN contribution	Roberta Presta
12.08.2016	IFS contribution	Thierry Bellet, Jean-Charles Bornard, Bertrand Richard
19.08.2016	UTO contribution	
23.08.2016	ENA contribution	Mathieu Magnaudet
24.08.2016	First integration of partners' contributions	Sonja Cornelsen (TWT)
25.08.2016	Final version for internal review	Sonja Cornelsen (TWT)
02.09.2016	Review from CRF	Fabio Tango, Daniel Prun
07.09.2019	Contributions from partners	OFF, SVN
08.09.2016	Final version	Cornelsen, Thurlings & All

Table of Contents

1 Introduction

HoliDes addresses the design and the development of Adaptive Cooperative Systems (AdCoS). These systems are adapting to the cognitive abilities of the human operator(s) and facilitate cooperation between humans and machines with the goal to enhance safety, to reduce the human error risk, and to better support human operators facing too complex or critical situations.
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. Further, the more the "intelligence" of the assistive systems increases regarding their adaptive abilities and a certain autonomy in decision making and actions implementation, the more complex is their technical verification and validation, and the more human-machine cooperation issues have to be carefully considered. Thus, a good Human Factors Design is essential to provide safety in these complex systems, especially concerning the human-machine interaction.
To better support the development of such AdCoS, which are able to harmoniously cooperate with humans and to adjust their assistance in accordance with the context of use, a Human Centred Design approach is crucial. The core objective is to guaranty safety in these complex systems, especially concerning the human-machine interaction, by considering both their technical *efficiency* (i.e. is the AdCoS able to efficiently perform the task it is in charge to implement or to support?) and their *effectiveness,* which is more related to end users' needs and expectations (i.e. how useful and adequate is the AdCoS regarding the task humans have to performed and the situational constraints they have to respect?).
To facilitate the development of Adaptive and Cooperative Systems, WP2 provides formal modelling languages, technics and tools that support the modelling of AdCoS used in WP6-9. Model-based approaches can be very helpful to manage system complexity, because models can be described on different levels of abstraction, focused on the relevant information in a structured way. One advantage of model-based approaches is that these models can be analysed in multiple ways, e.g. they can be checked for consistency, safety (formal methods) or efficiency. Moreover, some models and tools of WP2 are also used in WP3 to implement the adaptiveness of the AdCoS themselves. In addition, some of them are employed to guide empirical evaluation methods and design of AdCoS in WP5. Other models support their formal verification in WP4. Finally, the

models and languages developed in WP2 contribute to the common meta-model of the HF-RTP in WP1.

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, as well as the interactions between them. It facilitates 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, a simulation of the model allows for easier testing and thus an improvement of the product quality, while gaining shorter development at the same time.
MBD is widely used in e.g. aeronautics and space domain, but usage could be improved in all domains involved in the project. In the current industrial practice, there is only poor support for Human Centred Design and associated Human Factor Analysis, especially for adaptive and cooperative systems. In HoliDes, the MBD for AdCoS, including Human Factors aspects, was progressively tackled by defining and/or choosing a set of appropriate modelling languages and tools, allowing designers to model adaptation as well as human behaviors.
Figure 1 gives an overview of the three cycles of HoliDes, in which the modelling Languages and Tools were progressively developed and used.
Previous deliverables provided by WP2 presented the modelling languages, technics and tools in Cycle I, II and III.
In this deliverable, the following section will present an overview of the “common modelling framework” shared by the WP2 partners at a more transversal level.
Then, in section 3, the different Modelling Technics and Tools developed during the project by the WP2 teams, will be presented. All of them were already described in detailed in D2.5 and updated in D2.6. Thus, this sections aims at briefly recalling their description and then demonstrating their final status.
Finally, the last section will review which Requirements of the MTTs developed in WP2 have been achieved.

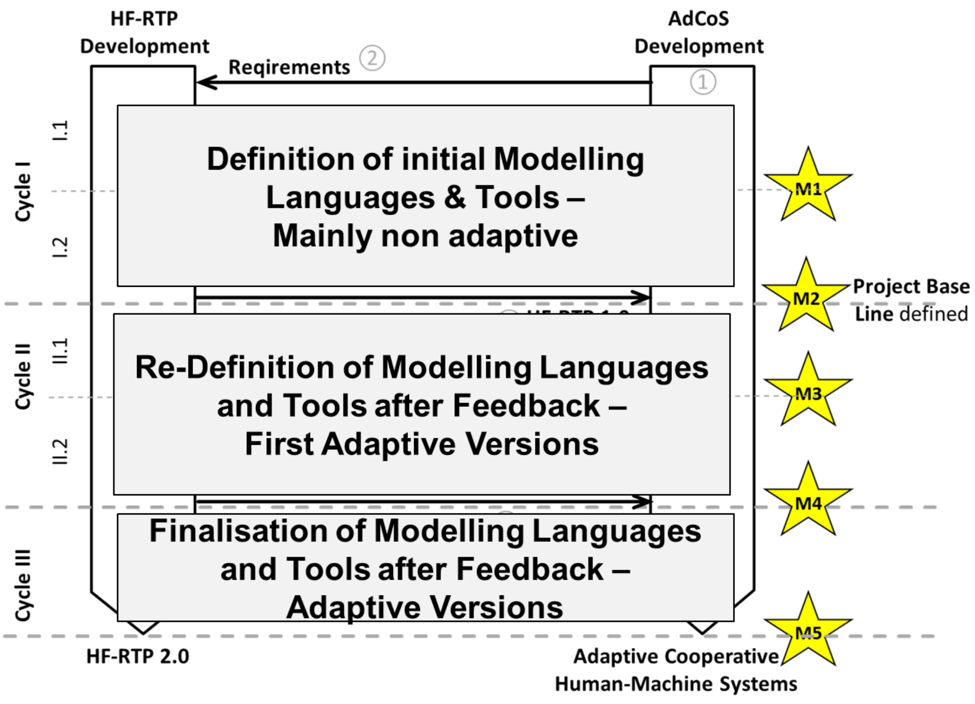

HF-RTP Development
AdCoS Development
Reqirements ②
①
Cycle I
I.1
I.2
Definition of initial Modelling Languages & Tools – Mainly non adaptive
M1
M2
Project Base Line defined
Cycle II
II.1
II.2
Re-Definition of Modelling Languages and Tools after Feedback – First Adaptive Versions
M3
M4
Cycle III
Finalisation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

This section presents an overview of the recent updates implemented in WP2 regarding Task Model, Resource Modelling Language, Human-Machine Cooperation Modelling, Human Operator Models and last but not least HMI Interaction Models.

All UML/ecore diagrams in this document follow a certain colour coding for the classes:

- Light Yellow: Normal Classes
- Grey: Abstract Classes
- Green: Enumerations
- Dark Cyan: Class from other Model, i.e. separate diagram

2.1 Task Model

The following section describes the final version of the Common Task Model for HoliDes, derived from the individual Task Models of the partners. There is no significant update from D2.5 and D2.6, despite the model has been split up into two packages, one for the task and goal hierarchies, and the second one for the rules package.

For a detailed introduction to Task Modelling and the partners individual Task Models, please refer to deliverable D2.5.

Figure 2 shows the task package of the Common Task Model of the HF-RTP V2.0, and Figure 3 shows the rule package of the Common Task Model.

Description of elements:

- TaskModel: Main entrance element that stores and links all model information
- Goal: A state that has to be achieved (e.g. a destination has been reached by car). A goal describes the WHAT in the task model.
- Objective: The objective describes WHY something has to be achieved.
- Task: A definite piece of work assigned to, or expected of an agent (human or machine). Describes HOW a Goal can be achieved. Tasks can be differentiated into:
    - UserTask: An internal cognitive activity, such as selecting a strategy to solve a problem

- InteractionTask: User actions that may result in immediate system feedback, such as editing a diagram
- SystemTask: Performed by the application itself, such as generating the results of a query
- AbstractTask: A task that has subtasks belonging to different categories, and thus cannot be allocated uniquely using the previous three categories

- Agent: A machine or human that is part of the AdCoS and is assigned at least one of the described tasks, details in the Cooperation Model
- Artefact: Link to the resources used by/needed for the task
- Relation: Describes the relation between two tasks, as described in the "RelationType" Enumeration, e.g. if they are running in parallel, if either of them is a choice, or if they are interleaving.

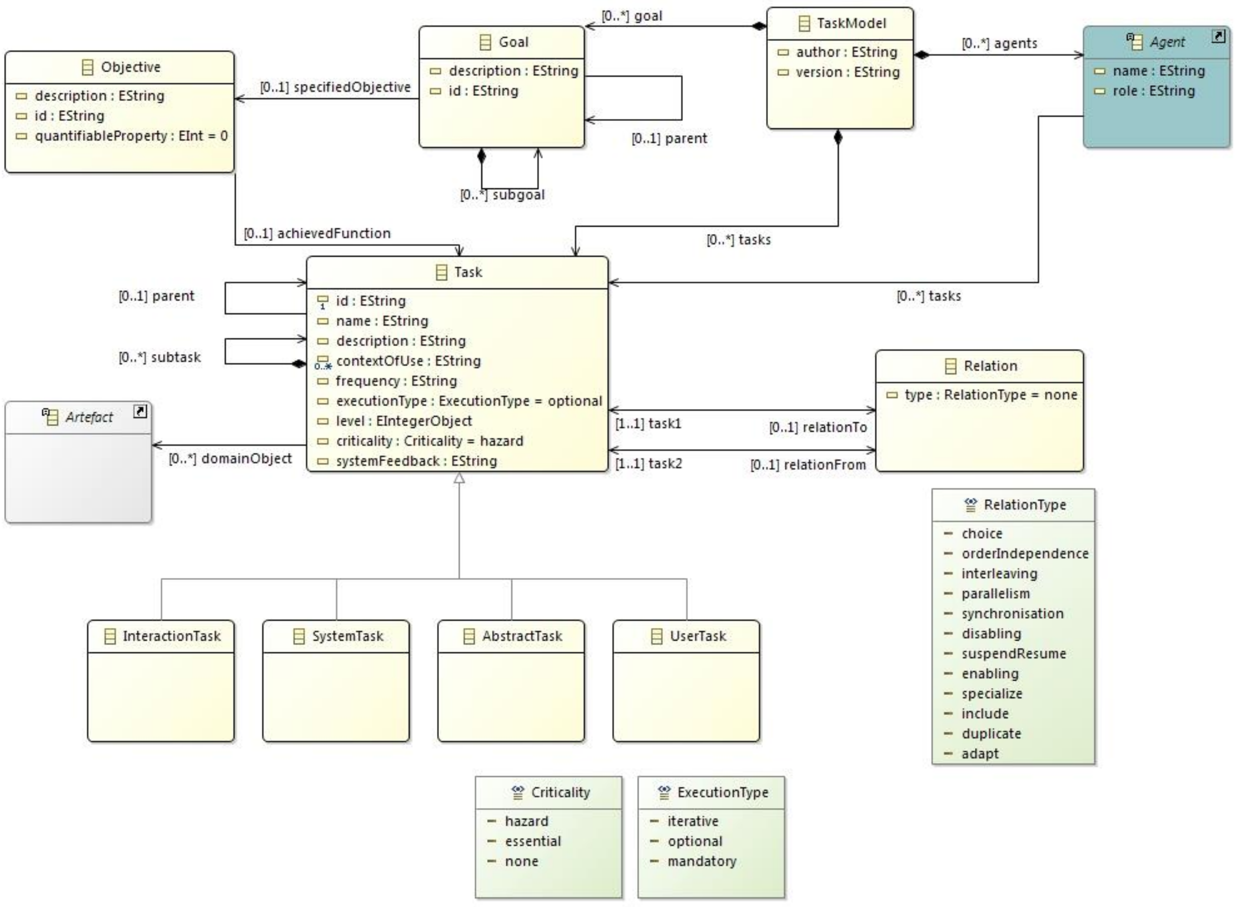


Figure 2: Common Task Model

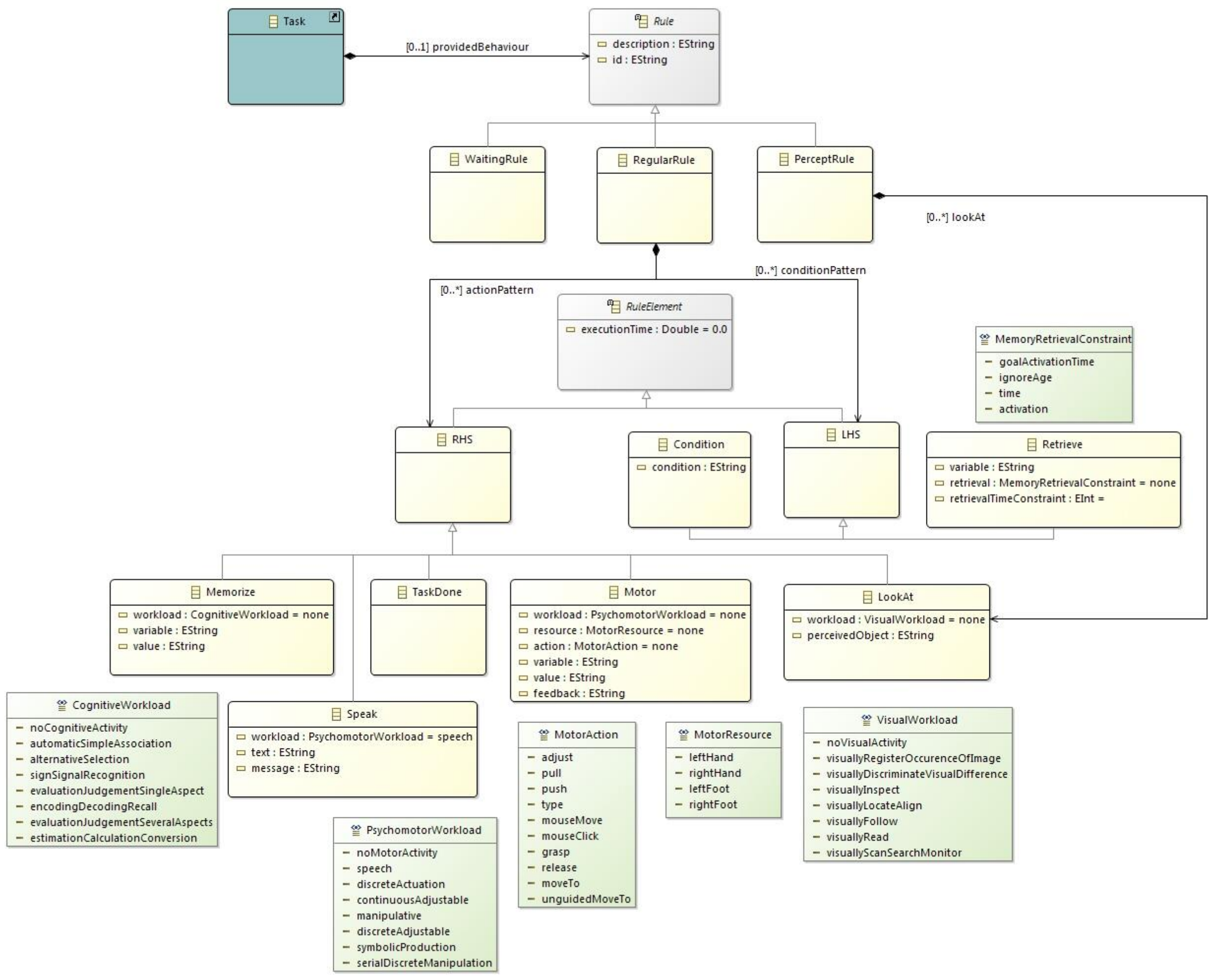


Figure 3: Rule Package - Common Task Model

Description of elements

- Rule: Rules can be used to describe on a very detailed level how a task is achieved. Each Rule belongs to a Task. There are 4 different types of rules:
    - RegularRule:
        - LHS: The Left Hand Side (LHS) of a rule describes the IF part
            - Condition: Boolean expression on the environment (Artefacts). The rule can be used if the condition is true.
            - Retrieve: Retrieves information from the memory that is needed for the evaluation of the Condition.
        - RHS: The Right Hand Side (RHS) of a rule describes the THEN part
            - TaskDone: A Task is done and removed

        - Speak: Say something to another agent
        - Motor: Perform an action with the hands or feet, e.g. typing, pressing a button, ...
        - Memorize: Store an information in memory for later retrieval
        - LookAt: Retrieve a needed information by visual perception (triggers eye-movement)
    - ReactiveRule: Rule not associated to a Task, in order to allow the description of reactive behaviour (i.e. triggers from outside, interruptions)
    - WaitingRule: Active waiting for something
    - PerceptRule: Triggers perception of visual information that is needed for the evaluation of a condition.

2.2 Resource Modelling Language

Figure 4 shows the ecore[2] Model of the final version of the HoliDes resource model. There is no update of the model from the previous version, as described in D2.5. For reference the following text from D2.5 has been added to describe the 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 T2.5.
- The **`HardwareArtefact`** class represents any resource that is hardware. This has been taken from DCoS-XML. Hardware is currently distinguished as **`DiscreteActuator`** (e.g. an on-off button, gear-shift), **`ContinuousActuator`** (e.g. the altitude selector of an aircraft's autopilot), **`Consumable`** or **`Sensor`**s.
- The **`EnvironmentalArtefact`** class represents any resource in the environment, e.g. a **`Space`**. This is currently not further defined.

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

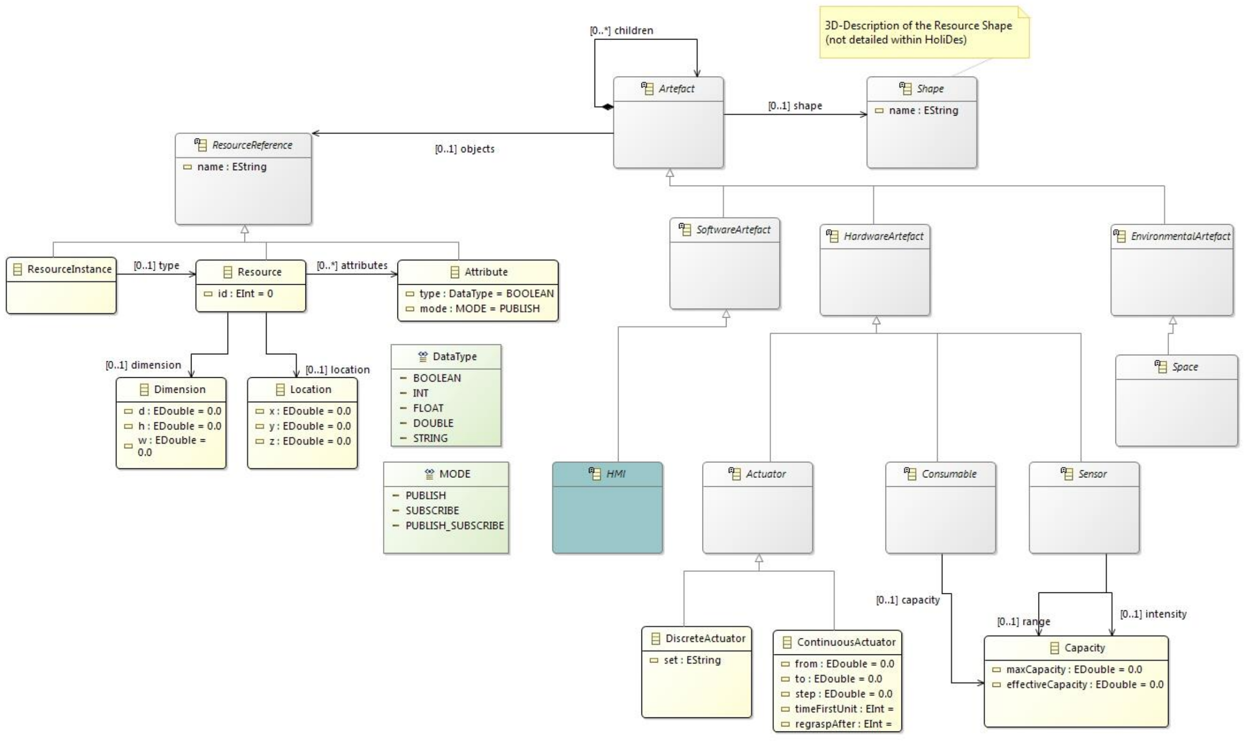


Figure 4: Resource Model

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

2.3 Human-Machine Cooperation Modelling

The Human Machine Cooperation model adopted by project HoliDes refers to Hoc's framework (Hoc, 2001), conceived on purpose for investigating different cooperation situations between human agents and machine agents from a human-centered perspective, i.e., by taking into account the cognitive aspects coming into play in the different cases.

The path towards the selection of the Hoc's model has been discussed in the previous WP2 deliverables (WP2.3, WP2.4, WP2.5, Section 2.3) and has been fully described in D2.6, Section 2.3, which is the reference document the reader is redirected to for further insights.

As a short summary, the framework envisions that the cooperation between the human part and the machine part of the human machine system is classified according to two dimensions: the level of cooperation, describing the level of the interference management activity in relation with the distance from the immediate action and with the elapsed time of such an activity, and the cooperation mode, in relation with the typology of the cooperation situation (Figure 5)[3].

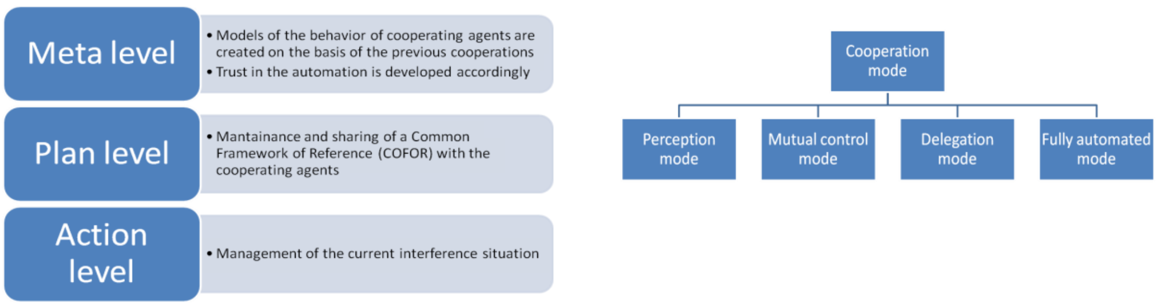

Meta level
• Models of the behavior of cooperating agents are created on the basis of the previous cooperations
• Trust in the automation is developed accordingly
Plan level
• Mantainance and sharing of a Common Framework of Reference (COFOR) with the cooperating agents
Action level
• Management of the current interference situation
Cooperation mode
Perception mode
Mutual control mode
Delegation mode
Fully automated mode

Figure 5. Levels of Cooperation (on the left) and Modes of Cooperation (on the right) in Hoc's framework

Mostly cooperation modes have been investigated and compared in literature.
In the *perception mode*, the machine part can be seen as an extension of the human perceptual system. It makes relevant information more easily accessible to the human, without providing interpretation of that information. In the *mutual control mode*, the human is in charge of controlling the machine and the machine is in charge of monitoring the human activities expressing criticism with respect to "risky" human behavior. Further specializations of the mutual control mode are defined. Hoc's framework, indeed, envisions different flavors for the mutual control mode. According to the assessment of the risks associated, to express the criticism the machine can, for example:

- Provide warnings ("warning mode")
- Provide suggestions ("action suggestion mode")
- Forbid the action ("limit mode"), or also

[3] Please see D2.6, Section 2.3, for details.

- Correct the action ("correction mode")

Navarro et al. proposed to distinguish between only two main categories for the mutual control mode: the "warning mode", including the aforementioned warning mode and action suggestion mode, and the "co-action" mode, including the limit and correction modes. The reason is that between these two main categories there is a clearer difference on the plane of human-machine cooperation: while the first two modes just give hints and suggestions to the human, the last two modes imply the human can no longer ignore the automation recommendation since it has an impact on the overall system behavior (Navarro, 2010).
The *function delegation* mode, finally, envisions that the human delegates part of the control task to the machine. When the delegation is total, the cooperation mode becomes the *fully automated mode.*

Pros and cons associated to the different cooperation modes have been explored and are still under evaluation, especially in the Aeronautics and Automotive domains, by means of different evaluation approaches.

Comprehensive reviews of cooperation modes analysis and evaluation, cited by following more specific works dedicated to ad-hoc cooperation implementations, can be found in Navarro (2010) and Hoc (2007). With particular reference to the automotive domain, Hoc (2007) reported results mainly based on experimental analysis. Lateral control in driving is the most considered goal of the human machine system under consideration. Suzuki (2003) and Navarro (2006) compared mutual control modes in this field. Both shown that a mutual control mode made by an auditory and haptic warning (a directed sound coming from the direction of the deviation and steering wheel vibration warnings) is efficient in reducing response time and maximum lateral deviation in critical situations, but the results in that terms in the case of road departures are less satisfactory than in lane departures (i.e., there is dependency from the context of the critical situation, the more stressful it is the lower are the benefit of the cooperation mode). Another cooperation mode examined in comparison with the previous one envisioned the use of the haptic channel to convey the suggested action to the driver, precisely by means of an asymmetric steering wheel vibration in the direction to be undertaken to mitigate the lack of centrality. This kind of cooperation mode resulted to be less efficient, since steering wheel torque is sometimes associated to lateral disturbances, thus causing unintended corrective steering reactions opposing the torque, canceling the action suggestion. However, sub-symbolic information provisioning, by means of

perceptual and sensorimotor interaction, is recommended to be further investigated for the communication between the human and the machine, because considered preferable in terms of processing efficiency by the human, as task analysis can prove in this dynamic contexts, by paying particular attention to the technology acceptance and training aspects.
In the review presented in Navarro (2010), human–machine cooperation modes are compared, besides reporting experimental analysis results, also by means of the performance comparison of the systems that instantiate the different approaches. Driving assistance systems representing examples of each considered cooperation modes are presented (e.g., vision enhancement systems as example of devices providing perception mode cooperation, lane keeping assistance systems and lane departure warning systems as examples of mutual control cooperation), and the metrics for the evaluation are on the basis of their impact on the driver behavior (such as the introduction of negative behavioral adaptation, intended as negative behaviors which may occur following the introduction of changes to the human machine system which were not expected or desired) and their impact on accident data, measured by accident reconstruction performed on the basis of a US accident database.
With reference to mutual control modes, conclusions highlighted that several studies indicate the potential safety benefits of warning modes, but mainly in simulation environments. Particular attention when adopting this cooperation option must be paid to the situation diagnosis (context assessment) before the actual onset of the warning. Indeed, at this level of human–machine cooperation, negative behavior adaptation can appear when there is an over-reliance of the human on the assistance system, with the consequent tendency for drivers to await the warning signal before adjusting their driving behavior.

Within the HoliDes project, as introduced and motivated in D2.6, Section 2.3, Hoc's cooperation framework has been used to represent in the context of the project evaluation activities two different cooperation models of the Adapted Assistance AdCoS (D9.9, Section 3.1.4), so that the evaluation results can be positioned among the aforementioned studies (see Figure 6 and Figure 7).
Indeed, the Adapted Assistance AdCoS HMI developed by REL[4] supports the driver by controlling his/her behaviour during the lane change considering the internal (driver's intention and distraction) and external (road and traffic conditions) context, implementing that way a "*mutual*

[4] The reader is redirected to D9.10, Section 3, for the description of the AdCoS and of the related evaluation activities.

control cooperation mode". In particular, the driver is provided visually with the adapted action to be performed, in terms of "keep your lane" and "change lane" indications. In case of risky condition (such as, for example, a lane change manoeuvre performing while a vehicle is approaching at a high speed from the rear), the adaptive activation of an auditory warning is provided in combination with a visual warning indicating the recommended action, to the aim of recall the driver attention during the manoeuvre (*auditory warning* + *visual warning*) (Figure 6).

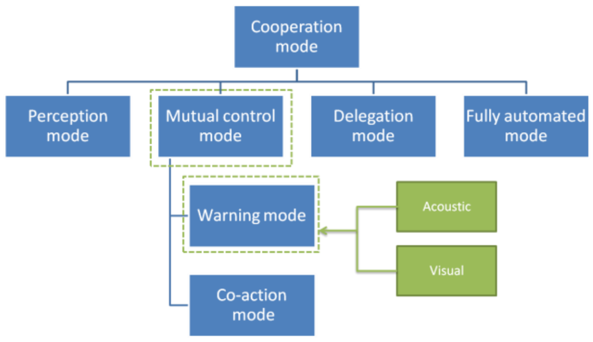


Figure 6. Cooperation model of the baseline AdCoS (acoustic and visual warning)

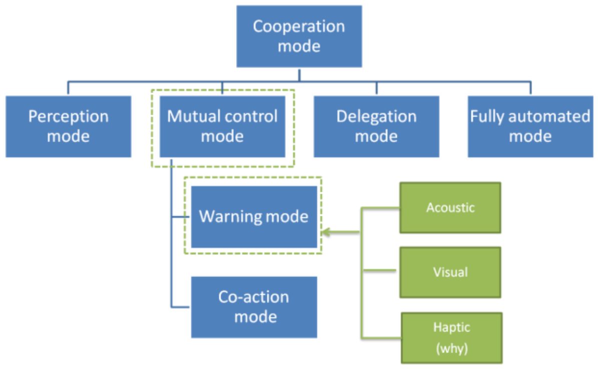


Figure 7. Cooperation model of the complete AdCoS (acoustic, visual and haptic warning explaining the why of the adaptation)

This cooperation model has been compared with a more sophisticated one where the communication of the "why" of the adaptation is envisioned by means of a haptic warning (auditory warning + visual warning + haptic warning). Directed haptic warnings are provided to the driver by means of a vibrating seat to indicate the provenance direction of the approaching vehicle that hinders the lane change maneuver from the rear, left and right, and by means of the vibration of the steering wheel to indicate a possible danger caused by the preceding vehicle (Figure 3).

With reference to the HMI, particular attention has been paid to the evaluation of subjective metrics related to the driver's perception of the cooperation modes[5]. HF RTP MTTs used in the evaluation activity have been experimental analysis and questionnaires about technology acceptance and usability, as well as ad-hoc questionnaires.
The evaluation activities have been designed according to experimental analysis principles, identifying as the baseline condition the cooperation model of the baseline AdCoS (Figure 6). The evaluation from the subjective perspective has been aimed at comparing, by means of specific questionnaires derived from well-known questionnaire models, the cognitive effort (perceived workload in terms of fatigue and distraction), perceived ease of use, usability, attitudes toward using and intention to use in both cases. Results show that both modes do not have significant differences in these terms, indicating that, even if the why haptic warning represents a cooperation mode the subjects are not used to, it is judged acceptable as other more familiar warning alarms. Besides, for the haptic warning, a dedicated questionnaire has been created for the assessment of the comprehensibility, distinguishability, perceptibility and effectiveness of the chosen signal. This specific questionnaire reveals that, even if the results of comprehensibility, distinguishability and perceptibility are satisfactory, the effectiveness, defined as the property of conveying the information about the direction of the danger, does not show the same positive results. The novelty of this functionality, unusual for a driver, has influenced the effectiveness for half of the participants. This result indicates that a learning phase might be needed, in order to let users get familiar with the adapted system and its safety mechanism, and subsequent tests to measure the improvements should follow.

Finally, the model has been specified in UML for integration into the common meta-model, as depicted in Figure 8, where it can be seen that the human-machine cooperation model linked to the Agent is characterized by a level of cooperation and by a cooperation mode, both expressed according to Hoc's framework.

[5] See D9.10 for more information about the evaluation activities.

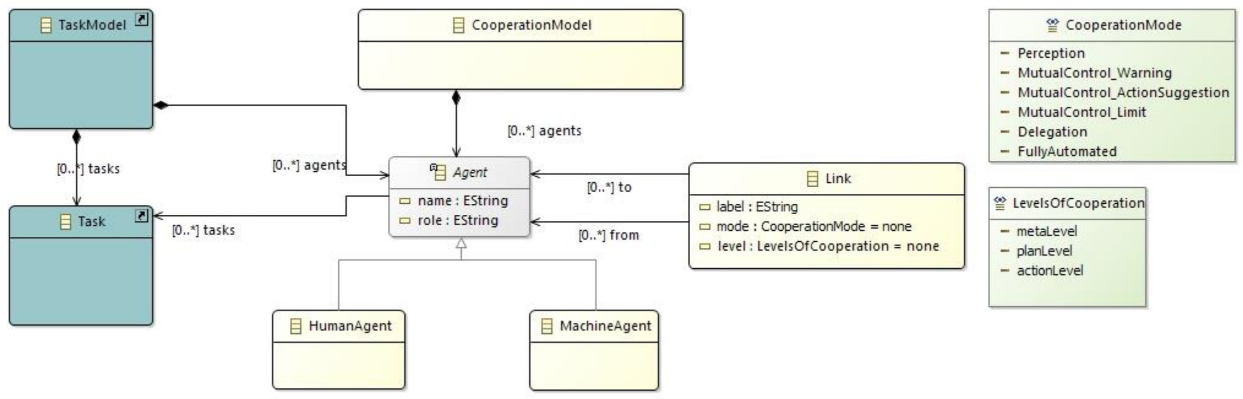


Figure 8: Common Cooperation Model

2.4 Human Operator Models

There are two types of "Human Operator Models" designed and developed in WP2: "Simulation Models", to be used as the Human Factor component into the HF-RTP, and "Monitoring Models", to be integrated in the AdCoS for supporting their adaptiveness.

Synthetically, ***Simulation Models*** (like CASCaS, COSMODRIVE and HEE) aim to represent human operators in a virtual way into the HF-RTP, in order to "simulate" or to "predict" their behaviours, their cognitive status, and/or their perceptive strategies when they have to perform a given task, in a given situational context (with or without an AdCoS). From these simulations or predictions, it is expected to virtually assess or predict, since the earliest stages of the AdCoS design process, performances, needs, difficulties, or potential human error risks of the future end-users of the AdCoS. Beyond the behavioural performance of human operators, the objective is also to assess, to predict or to simulate their cognitive processes and their mental states (like *Situation awareness* or *distraction status*), underlying and explaining their performances.

Regarding ***Monitoring Models*** (like the MPDN Co-pilot, BAD MoB, Cognitive Distraction Classifier, or Pilot Pattern Classifier), the aim is not to virtually "simulate" the human operators in the HF-RTP, but to "model" them into the AdCoS themselves, in order to "analyse" the activities of real humans, to assess their mental state, and/or to evaluate their performance, from an "external observer" point of view (i.e. as monitoring functions). Synthetically, the aim is to provide on-line or off-line diagnosis concerning humans' perceptive and/or cognitive states (like level of distraction or fatigue, for instance), or to evaluate the quality of their performance (e.g. is the observed behaviour adequate or not, according to the situational context and the requirements of the task they have to perform), in order to support accordingly the adaptive and cooperative abilities of the AdCoS.

The different Human Operator Models (of one of these 2 types) developed in WP2 will be respectively presented in a detailed way in section 3.

2.5 HMI Interaction Models

2.5.1 The problem of HMI programming

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

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

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

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

Traditional programming languages have received extensions to support the execution of interactive software. For example, waiting functions support execution control by external inputs, and threads support parallel execution of actions. With these extensions, they theoretically support the development of interactive software. However, the increase of possible inputs, states and combinations of components dramatically increases the number of possible executions of a given application. If this multiplicity of possible executions is programmed using the usual control structures, software complexity increases: any modification of the program behaviour requires changes in multiple components, thus restraining the ability to make choices after the initial design phase.
Along with this increase in software complexity, the interoperability of software components tends to decrease, and software development and validation become long, costly and prone to errors. It also becomes difficult to analyse the properties of software at the appropriate level of abstraction, and only certain classes of interactive software can undergo the software certification processes required in some industrial fields. It also becomes difficult to design programming tools that facilitate software development, because there are no visual representations that appropriately capture the structure of software.

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

Various software patterns have been proposed to curb software complexity by organizing software components according to their roles and defining how they can be combined. For example, with the Model-

View-Controller pattern, application components are made of three sub-components that are respectively in charge of managing the data and the computation, visualizing the data, and managing user input. The Presentation-Abstraction-Control and Model-View-View Model patterns have similar structures. Extended scene graphs are another class of patterns, derived from graphical scene graphs, in which various kinds of non-graphical software components can be added as nodes of the graph, so as to align the software architecture of the application on its graphical structure.

Other patterns have been proposed to organize control flows in interactive software, and compensate the limitations of control structures provided by programming languages. For example, (Harel 1987) proposes Statecharts, hierarchical state machine components that can be combined to describe interactive systems. (Myers 1990) describes a state machine component that can be adapted to program interaction in various kinds of software components. Transitions between states are performed at the occurrence of certain events, and the appearance and behaviour of software depends on said state of software. (Dragicevic 2004) proposes a data-flow system that can be used to program input management. (Nigay 1993) describes a multimodal fusion pattern for combining events and states from multiple inputs. (Sottet 2007) proposes a pattern for managing the adaptation of software to changes in the computing platform and the execution context.

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

Partial solutions have been proposed to make these software patterns interoperable. For example, (Chatty 1994) proposes a method for combining state machines and data flows, in which the configuration of data flows changes when state changes. (Appert 2008) proposes another method for combining state machines and data flows, using Java code to perform the adaptation. (Elliot 1997) proposes Functional Reactive Programming, an alteration of the execution semantics of functional languages that allows exploiting the same syntax for expressing both traditional computation and data flows. (Chatty 2004) discloses an application of extended scene graphs for assembling graphics and heterogeneous behaviour components in a homogeneous fashion.

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

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

Synchronous data flow languages have been created to support the creation of interactive software such as automatic control systems. (Halbwachs 1991) describes a synchronous dataflow language, LUSTRE. Extensions to LUSTRE have been developed to implement user interfaces. In LUSTRE inputs are used for controlling data flows. In addition, LUSTRE code can be used to define state machines. However, the interoperability between state machines and data flows in LUSTRE is limited as in previously described solutions. In addition, it is very difficult to replace one data flow with another, once it is defined. The definition of new control structures is not supported.

For HoliDes, the description of the HMI Interaction model together with the structure of the model is shown in Figure 9.

2.5.2 A new event-based approach

Remark: this part has been subject of an official communication. Refer to (Chatty 2016) for more information.

"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).

An architecture of reactive components

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

Programmers create interactive programs by instantiating and assembling software components, and connecting them to hardware components. The *djnn* environment provides some basic software components to this purpose: components that support user interaction, components for data representation, computation-oriented components, components that encapsulate pre-existing code written in another language, components aimed at assembling and connecting other components.

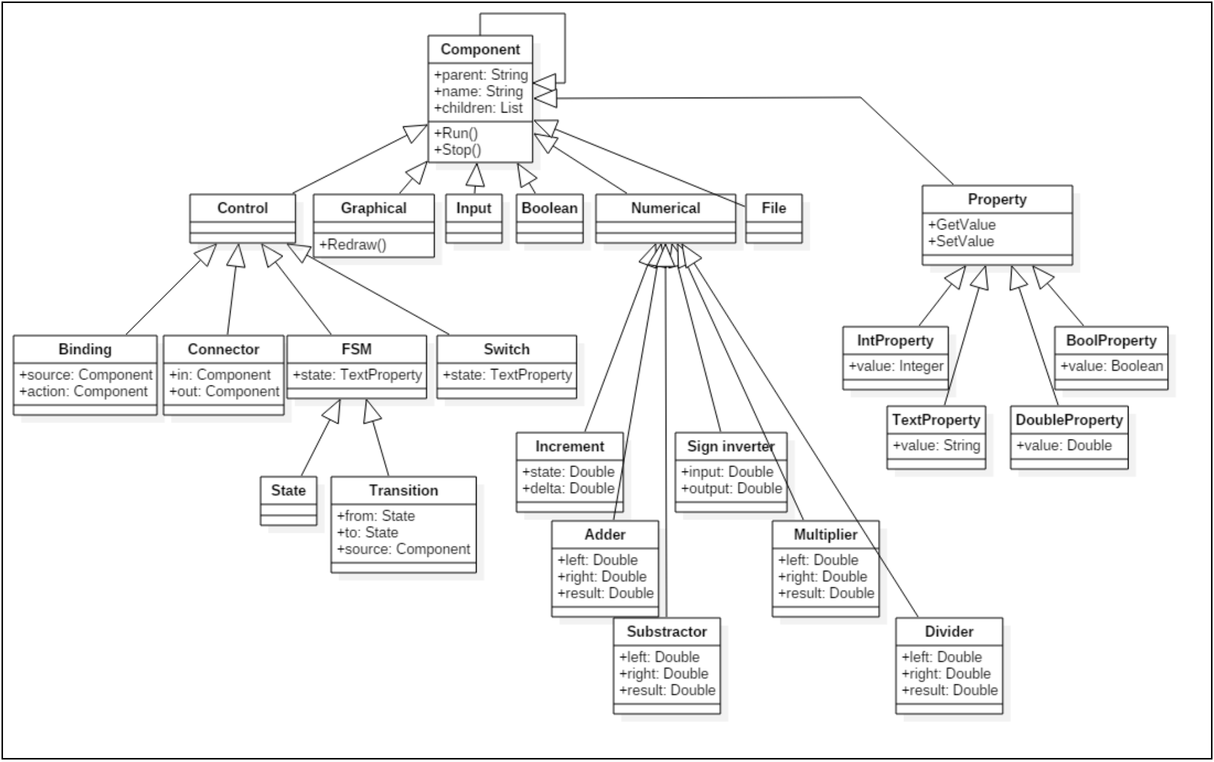


Figure 9: Simplified UML representation of the component hierarchy

We describe hereafter some of these basic components.

<u>Control oriented components</u>

djnn provides the **control structures** that have been introduced for interactive software in the last decades, as well as traditional computation-oriented control structures. The *djnn* fundamental control primitive is called “**binding**”. A binding is a component that creates a coupling between two existing components. If there is a binding between components C1 and C2, then whenever C1 is activated, C2 is activated (C1 is called trigger and C2 is called action). A binding can be interpreted as a transfer of control, like a function call in functional programming or a callback in user interface programming.
Figure 10 shows simple examples of bindings definition.

```
# beeping at each clock tick
binding (myclock, beep)

# controlling an animation with a mouse button
binding (mouse/left/press, animation/start)
binding (mouse/left/release, animation/stop)
```

```
# quitting the application upon a button press
binding (quitbutton/trigger, application/quit)
```

Figure 10: Examples of bindings definitions in djnn

Bindings are used to define a set of additional control structures required to describe interactive software:

- **Switches**, which activate one among several components depending on the value of their state.
- **Connectors**, which ensure that any modification of their input value are propagated to their output values;
- **Watcher** (allow to connect C1 and C2 to C3 where C3 is activated only when C1 and C2 are synchronously activated)
- **Composite components**, which propagate their activation to their sub-components;
- **Iterators**, which activate other components in a given order;
- **Tests**, which activate another component only when they are activated and when a boolean value is true;

Figure 11 shows example of control oriented components definition.

```
# ensure that rectangle rect1 will move with
# the mouse.
connector (mouse/position/x, rect1/position/x)
connector (mouse/position/y, rect1/position/y)

# m is performed when input1 and anput2 are
# simultaneously activated
multiplication m (input1, input2, output)
watcher (input1, input2, m)
```

Figure 11: examples of derived control structure definitions in djnn

FSM are one of the most used control structures for describing user interfaces with *djnn*. They contain other components named states and transitions. Transitions are bindings between two states (named origin and destination). A transition is active only when its origin is active. It behaves as a binding with a default action: changing the current state of the FSM to its destination state. Therefore, the transitions define the inputs of the state machine: the state evolves on the sequence of activation of the triggers of the transitions, and ignores events that do not match the current state. Figure 12 shows the internal behaviour of a software button designed for use with a mouse: the *djnn* code above implement the FSM shown at the bottom, where "r" is the image of the button (a rectangle component).

```
component mybutton {
 rectangle r (0, 0, 100, 50)
 fsm f {
   state idle, pressed, out
   transition press(idle, r/press, pressed)
   transition trigger(pressed, r/release, idle)
   transition leave (pressed, r/leave, out)
   transition enter (out, r/enter, pressed)
 }
}
```

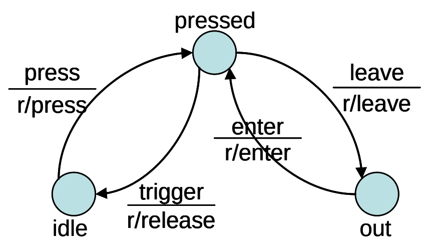


Figure 12: Example of a FSM definition

Programmers can extend this basic set by assembling available components to produce new control structures dedicated to their own needs, for instance control structures dedicated to software adaptation.

Components for input/output handling

In addition to these control structures, *djnn* comes with a collection of basic types of components dedicated to handle user interfaces:

- graphical oriented components (rectangle, ellipse, circle, text, line, polygons, opacity, gradient, rotation, translation, …)
- input components (mouse, multi-touch, sensors, etc.),

Other components

Other components allow the programmer to describe additional elements:

- file management components (load and save a graphical file)
- numerical oriented components: increment, addition, substractor, multiplier, divisor, …
- Boolean oriented component (and, or, xor, not, comparator…)
- Property: they are component that can manage a piece of information (integer, Boolean, text…) and provide access to other components.
- etc.

Components 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 rectangle component comes with some basic sub-components:
    - `x`: abscissa of the rectangle
    - `y`: ordinate of the rectangle
    - `width`: width of the rectangle
    - `height`: height of the rectangle

- `rx`: horizontal size of the round corners
- `ry`: vertical size of the round corners
- `press`: mouse pressed event. The press element has 2 children: `x` and `y`
- `release`: mouse released event
- `move`: mouse moved event. The move element has 2 children: `x` and `y`
- `enter`: mouse enter event
- `leave`: 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 Finite State Machine (FSM) is made of several bindings etc. The designer can explicitly manage this tree-oriented architecture.
- Transversal connection: all the available components can be connected by control primitives, whatever is their place in the tree of components. For example, a binding can connect a mouse press to a rectangle horizontal position.

Combining FSMs by coupling their transitions, or by controlling the activation of one by a state or a transition of another, makes it possible to create complex behaviours (Figure 13). It also makes it easier to structure applications as collections of reusable components.

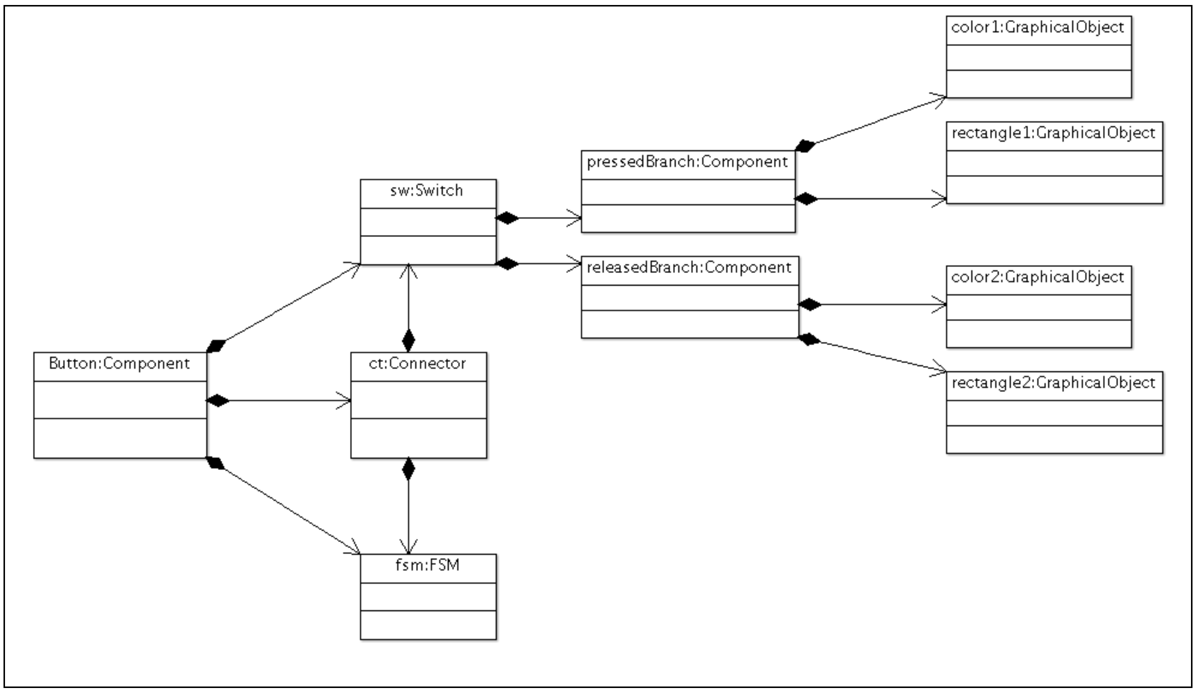


Figure 13: UML representation of a simple djnn button

2.5.3 A model of djnn components

In the context of Holides project, and specially WP2 objectives, a model of *djnn* components has been defined. An abstract syntax and a grammar for *djnn* have been defined through various XML schemas.

Figure 14 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 14: XSD definition of two djnn components (binding and FSM)

The main advantages provided by these definitions are:

- Definition of a well-defined model for *djnn*: illicit constructs using the language are easily and automatically 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 is possible to investigate such properties with dedicated tools. These tools are described in WP4 D4.7 document.

2.6 Training Models

The training model has been specifically derived for the WP7 Training AdCoS, where the model is evaluated, and has been generalized for the final version of the training model, by removing aircraft specific elements and naming.

Figure 15 shows the final model for training application. There are no significant changes in the model compared to D2.6. The only change is that a rating model has been added to the standard, allowing the trainers to rate the trained elements during the course. Figure 16 shows the rating model.

In order to have a complete description with the model, the following text is taken from D2.6 to explain the training model:

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

In addition, the procedure is assigned to a (journey) phase, in which the procedure is usually applied. For each **`JourneyPhase`**, one can specify which elements must be trained in this phase, i.e. often certain procedures are only applied in a certain phase (e.g. starting the aircraft engines on ground during pre-flight parking), and some malfunctions can only occur during a certain phase (e.g. the ignition can only fail when engines are started). In order to express this, the procedures can be specified as part of a **`Sequence`** (all procedures have to be trained) or a **`Choice`** (per session only one element is trained, but all have to be trained during the complete training). More details on that are described in the section 3.3 of the SyllabusManager in D2.6.

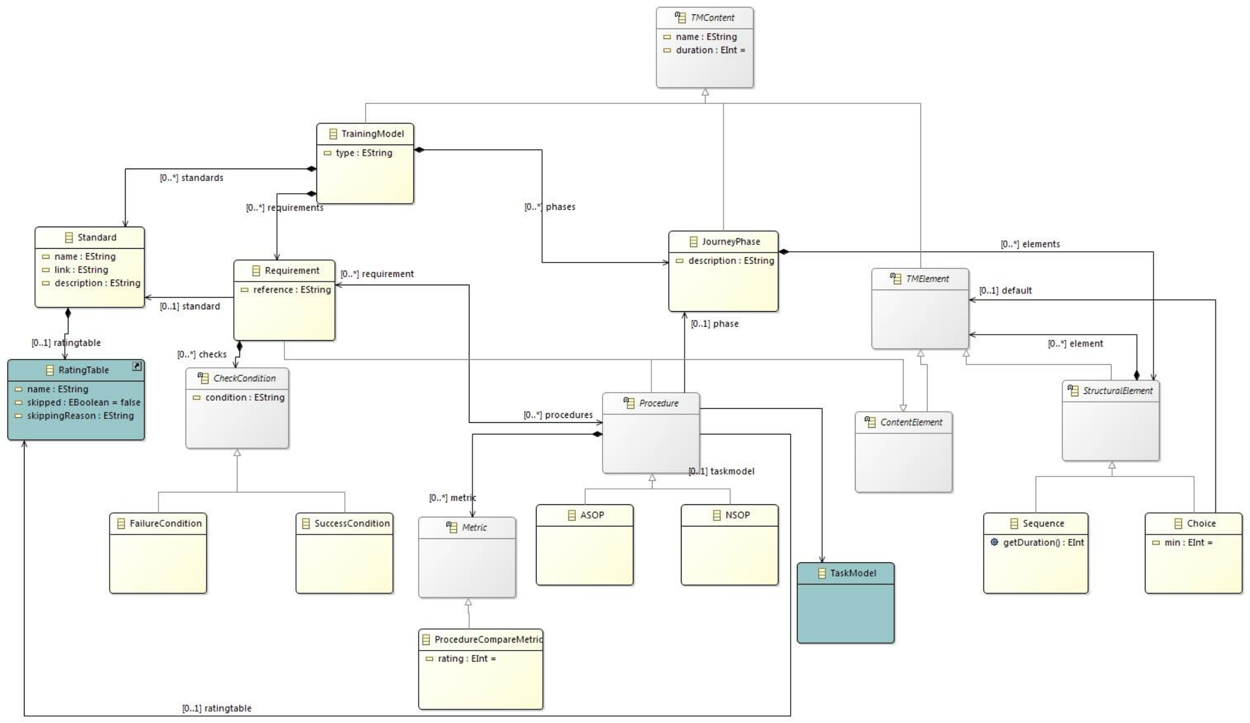


Figure 15: Common Training Model

As explained above, the training model has been extended with the rating model. For that, the **`Standard`** class has several containment references to a **`RatingTable`**, describing all possible RatingTables that can be used in this model. Each Procedure is associated (reference) with one of these Tables, thus allowing re-use of certain tables in different procedures.

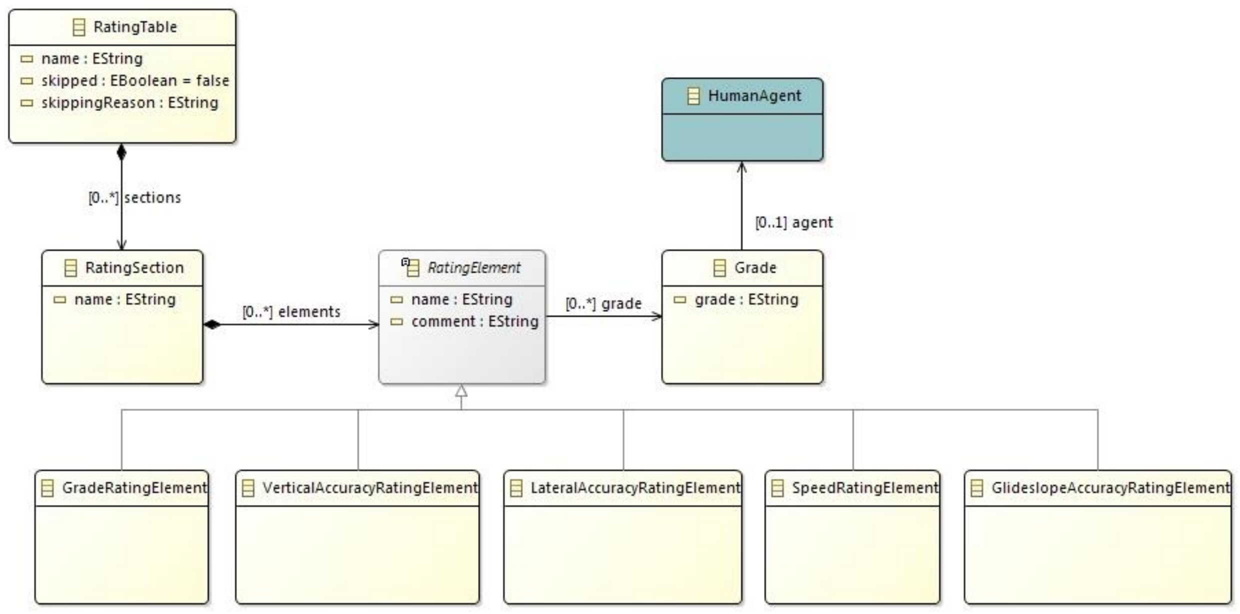
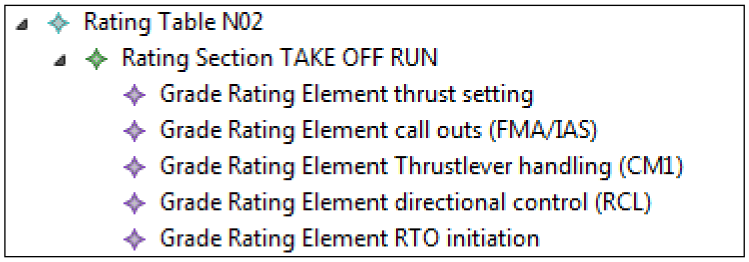


Figure 16: Rating Model for Common Training Model

A **`RatingTable`** contains several sections grouping **`RatingElements`** together. A RatingElement has a name explaining what to rate, allow to store a comment, and has a list of grades, for each trainee (**`HumanAgent`**). The grading scheme (i.e. from 1 to 6) is defined by the different sub-types of the RatingElement:

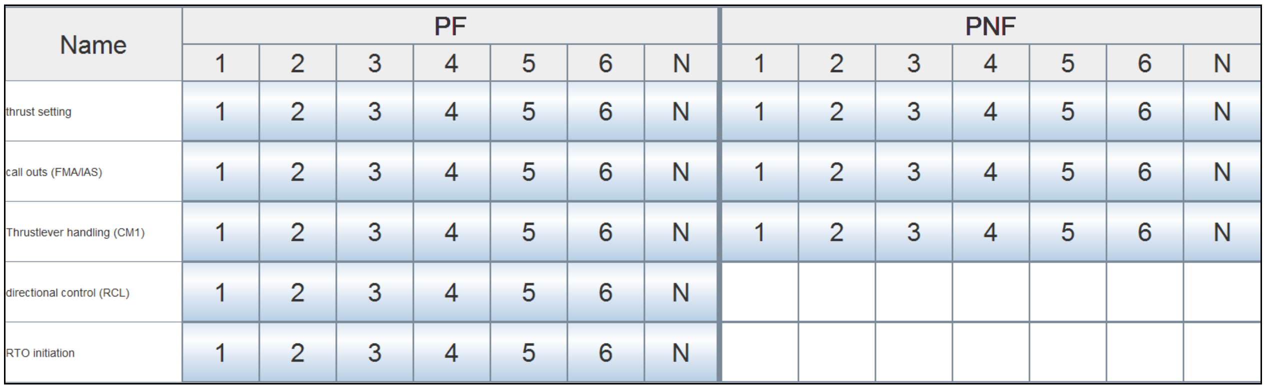
- **`GradeRatingElement`**: Rating from 1 to 6 and N for "Not Observed"
- **`LateralAccuracyRatingElement`**: 5+,5,2,2,1,0,N (degree deviation)
- **`VerticalAccuracyRatingElement`**: 200+,100+,50+,40+,30+,10+,N (feet deviation)
- **`SpeedRatingElement`**: 15+,15,10,7,5,2,N (knots deviation)
- **`GlideslopeAccuracyRatingElement`**: 2.0+,2,1.5,1,0.5,0.2,N (dots deviation)

An example RatingTable from the aircraft domain (take off run), is shown in Figure 17, and the constructed view in the SyllabusManager from this model is shown in Figure 18.



- Rating Table N02
  - Rating Section TAKE OFF RUN
    - Grade Rating Element thrust setting
    - Grade Rating Element call outs (FMA/IAS)
    - Grade Rating Element Thrustlever handling (CM1)
    - Grade Rating Element directional control (RCL)
    - Grade Rating Element RTO initiation

Figure 17: Example Model: TAKE OFF RUN from aeronautics domain



Name	PF							PNF						
	1	2	3	4	5	6	N	1	2	3	4	5	6	N
thrust setting	1	2	3	4	5	6	N	1	2	3	4	5	6	N
call outs (FMA/IAS)	1	2	3	4	5	6	N	1	2	3	4	5	6	N
Thrustlever handling (CM1)	1	2	3	4	5	6	N	1	2	3	4	5	6	N
directional control (RCL)	1	2	3	4	5	6	N							
RTO initiation	1	2	3	4	5	6	N							

Figure 18: Example Rating Table for Aeronautics Domain

3 Modelling Techniques and Tools V2.0

This section is dedicated to the presentation of the different Modelling Technics and Tools developed by WP2 partners during the project. All of them were described in detail in previous deliverables, including D2.6. This section is mainly focused on their current status according to their recent update implemented during the last period.
Table 1 gives an overview on the tools and their application in the AdCoS domains:

Table 1: Tools and techniques and their application in the domains of Health, Aeronautics, Control Room, Automotive and other domains outside HoliDes. TTs that *can* be used across domains (AD), are already used cross-domain (CD), that are used cross-companies (*CC)* or cannot be used cross-domains but influenced other TTs (I).

				health					aeronautics		control room		automotive			
Tool	**Extended Name / Description**	**Used Model**	**Prov ider**	**PHI**	**UMC**	**PHI**	**ATO**	**IGS**	**HON**	**TRS**	**CAS**	**IRN**	**CRF**	**IAS**	**TAK**	**IFS**
MagicPED	Procedure Editor extension of MagicDraw UML Tool	Task Model	OFF							CD						
CASCaS	Cognitive Architecture for Safety Critical Task Simulation	Cognitive Model	OFF												CD	
HEE	Human Efficiency Evaluator	Task Model, Cognitive Model (CASCaS)	OFF			CD									CD	
SyllabusManager	Tool for creation of adapted training syllabi	Task Model, Training Model	OFF							CD CC						
COSMODRIVE	Cognitive Simulation Model of the car Driver	Cognitive Model	IFS													
GreatSPN for MDPN	Editor for MDPN used as virtual co-pilot	Petri Net Model	UTO													CC
DIR (former BAD-MoB)	Driver Intention Recognition models	Human Behaviour Model	OFF										CC			
Cognitive Distraction Classifier	Model of human cognitive distraction	Applied Cognitive Model	TWT						CD				AD	CC	CC	AD
djnn	HMI model editor and execution	UI / Interaction Model	ENA													CC CD
Visual Distraction Classifier	Driver Distraction Classifier	Human Behaviour Model	UTO													
Pilot Pattern Classifier	Pilot Pattern Classifier	Human Behaviour Model	TEC						CD				AD	AD	AD	AD

For modelling Techniques and Tools (TT), integration through OSLC is only planned for lifecycle tools, but not for non-lifecycle TTs. Table 2 gives an overview which TTs are lifecycle and which are non-lifecycle tools.

Table 2: List of lifecycle and non-lifecycle TTs

Lifecycle TT	Non-lifecycle TT
MagicPED	Cognitive Distraction Classifier (CDC)
SyllabusManager	CASCaS
Human Efficiency Evaluator (HEE)	Driver Intention Recognition (DIR) models (former BAD-MoB)
	djnn
	Pilot Pattern Classifier (PPC)

3.1 Magic PED (OFF)

MagicPED is the Task Editor of OFF, which is based on the commercial UML tool MagicDraw by NoMagic Inc[6]. Using an UML tool has the benefit that the potential users (e.g. Software engineers) can stay in their tool world for developing the system. MagicPED consists of two parts:

1. The UML editor MagicDraw itself. MagicDraw provides a full featured UML editor, and provides, next to model validation, an API for extending MagicDraw via plugins, and an additional TeamServer, which allows to cooperatively work on models.
2. A package by OFF with the UML profile for the task models and a set of plugins, extending the editor of MagicDraw.

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

As there are no significant changes to the previous version, a more detailed description is omitted in this deliverable. Details on MagicPED can be found in D2.6, section 3.1, and Annex II of D2.4 (MagicPED Handbook).

[6] http://www.nomagic.com/products/magicdraw.html

3.2 Human Efficiency Evaluator (OFF)

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. The Human Efficiency Evaluator (HEE) overcomes these drawbacks, by providing an easy-toö-use tool that supports evaluation in early design phases, by prediction of task performance, operators' workload, 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.

Typical design questions that can be answered 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?

Since D2.6, there have been no significant updates on the model and the editor of HEE, thus a detailed description is not added here. Details can be found in D2.6, section 3.2, and in D2.5, section 3.2.2 a state of the art review can be found.
In the last year, the HEE has been applied by the partners as part of the HF-RTP in two different domains: Healthcare and Automotive. Within the Health domain it was used for evaluation of the 3D-Acquisition AdCoS by PHI and also in the OpenEHR use case by ATO. Details on the results are described in D6.9. Recently, the HEE has been also been used by the DLR to evaluate an urban automotive scenario (traffic light assistance) and currently TAK is applying the HEE to evaluate their use case about an adaptive HMI in lane change and overtaking scenarios. The former results have been accepted to be published at the Automotive UI conference 2016 and the latter results are currently analysed and are planned to be reported for D9.10

3.3 SyllabusManager (OFF, TRS)

OFF and TRS decided to rename the TrainingManager to SyllabusManager. In the previous deliverables, this tool can therefore be found in under the name TrainingManager, while in this deliverable we use SyllabusManager.

3.3.1 MTT Description

The SyllabusManager is developed by OFF in cooperation with TRS within HoliDes. Objective is to develop a tool, allowing modelling of all aspects of a transition training (i.e. training from one aircraft type to another), in terms of

- Procedures to be trained by the trainee (Standard Operating Procedures (SOPs))
- Flight Crew Licensing Requirement (FCLRs; coming from Regulations)
- Training syllabi, including flight phases, scenarios, …
- Learning Knowledge
- Adaptation of the syllabus to previous knowledge (i.e. flown aircrafts, previous licences).

The SyllabusManager takes the SOPs from two different aircrafts, and allows the trainer to create new training syllabi. For the adaptation, the SyllabusManager provides a categorisation of the different procedures, which are based on the differences of the SOPs. In addition, a "novelty score" of each training element is calculated, that depends on

- the category of difference (the more different to previous knowledge, the more novelty is added)
- the number of repetitions this element has been already trained (during the course)

The SyllabusManager has been successfully applied in the aeronautics domain in Use Case 7.2 "Adaptive Flight Crew Simulator Transition Training".

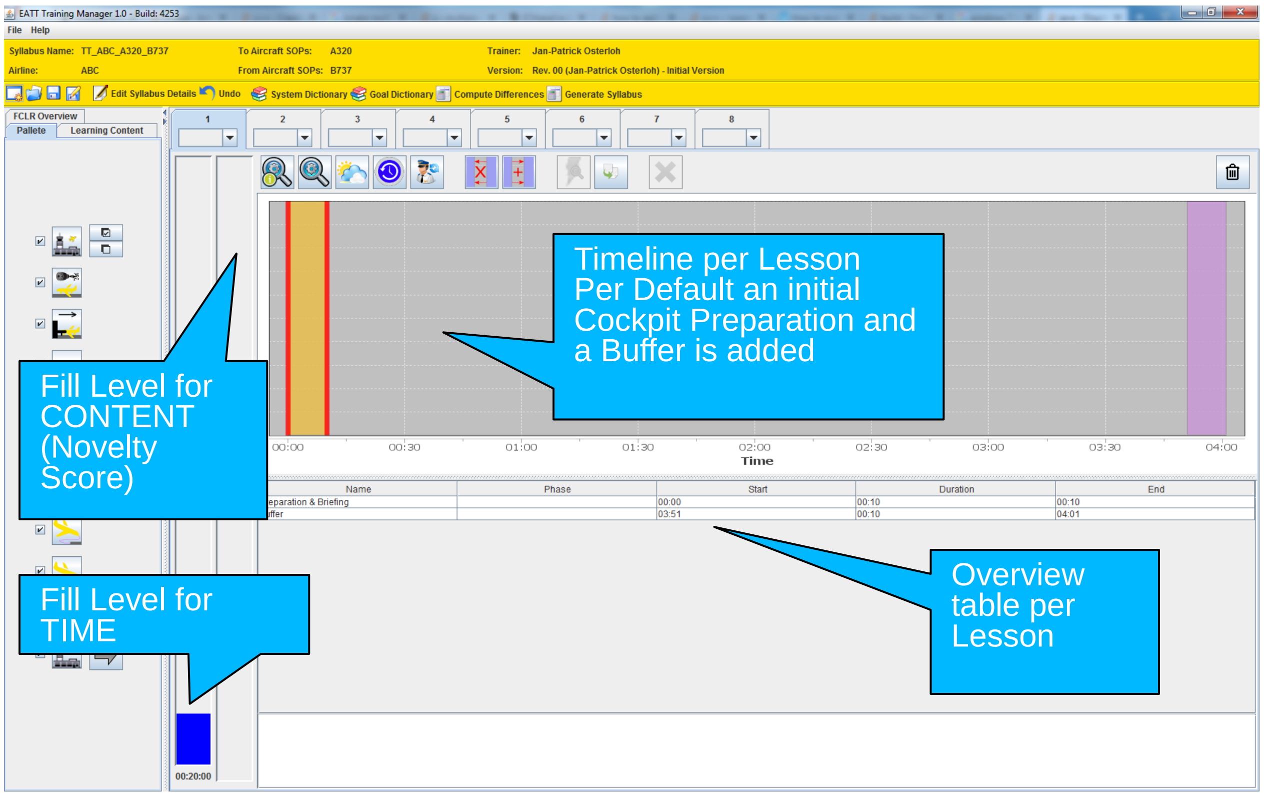
3.3.2 State of the art

Please refer to deliverable D2.5, section 3.3.2 for a state of the art review.

3.3.3 Current status

Since D2.6, several improvements have been implemented in the SyllabusManager. Main changes in comparison to the D2.6 descriptions are:

- Editing is now done in one step, i.e. the training elements can be added directly to the timeline of a lesson by drag and drop.
- Export to Word has been added (as printout for students and trainer)
- A palette for quick insertion of training elements has been added.
- The configuration of the elements that are inserted has been replaced with a wizard.


EATT Training Manager 1.0 - Build: 4253

Syllabus Name: TT_ABC_A320_B737 | To Aircraft SOPs: A320 | Trainer: Jan-Patrick Osterloh
Airline: ABC | From Aircraft SOPs: B737 | Version: Rev. 00 (Jan-Patrick Osterloh) - Initial Version

Name	Phase	Start	Duration	End
eparation & Briefing		00:00	00:10	00:10
uffer		03:51	00:10	04:01

Figure 19: Screenshot Main Window of SyllabusManager

A more detailed description of the tool can be found in Annex II, the Handbook of the SyllabusManager.

3.3.4 Integration status

Integration in the HF-RTP is completed, see also D7.9.

3.4 CASCaS (OFF)

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.

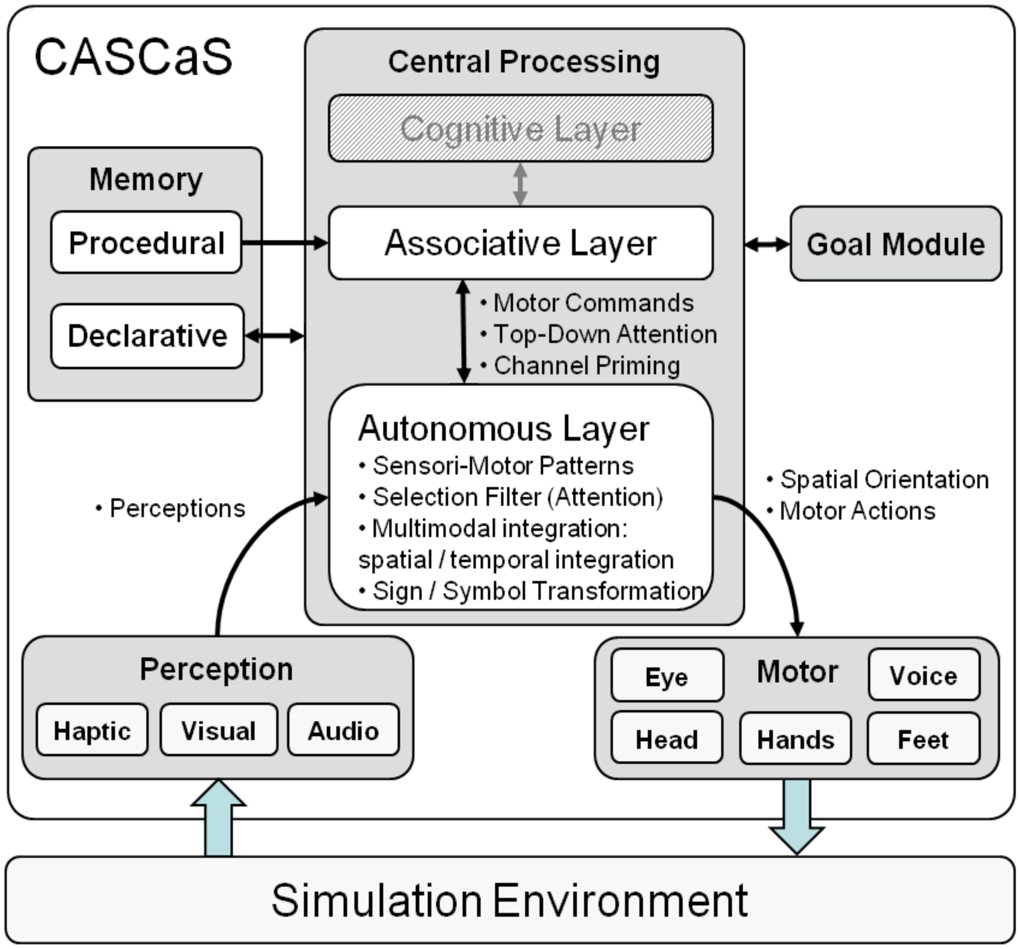


Figure 20: Structure of the cognitive architecture CASCaS with all internal components and the major data flows

Figure 20 shows the current architecture of the cognitive model with all its components. Basically, the architecture consists of 5 components: a *Goal Module,* which stores the intentions of the model (what it wants to do next). The *Central Processing* is subdivided into three different layers: the *cognitive layer* which can be used to model problem solving, the *associative layer* executes learned action plans and the *autonomous layer* simulates highly learned behaviour.

CASCaS is used as simulation of human behaviour in the Human Efficiency Evaluator. Please refer to D2.6 section 3.2 for more details on the HEE, and D2.6 section 3.4 for more details on CASCaS.

3.5 COSMODRIVE (IFS)

3.5.1 State of the art

COSMODRIVE (for Cognitive Simulation Model of the car Driver) is a long-term research program of IFSTTAR (Bellet et al, 2007, 2009, 2012) aiming to provide a computational simulation model of car drivers. The general objective is to virtually simulate the human drivers' perceptive and cognitive functions implemented when driving a car, through an iterative "Perception-Cognition-Action" regulation loop.

However, a totally new version of this model has been specifically developed for HoliDes project, in order to be interfaced with other HoliDes MTTs (like RTMaps, Pro-SiVIC or Djin), and then to be used in WP4 to support the Virtual Human Centred Design (V-HCD) of future AdCoS in charge to support human operators in automotive domain (WP9).

Regarding more specifically the "HF-RTP approach" in HoliDes, COSMODRIVE plays the role of a "Human Factor" component interacting with a "RT-Platform" based on RTMaps and Pro-SIVIC software (provided by INT and CIV), and then proving an instance of a tailored HF-RTP for supporting AdCoS design and test in WP9: the V-HCD platform (described in D4.6/4.7). From this integrative platform, it is possible to generate dynamic simulations of a car driver (simulated with COSMODRIVE), interacting with a virtual road environment (simulated with Pro-SIVIC), through actions on a virtual car (simulated with Pro-SIVIC), equipped with a Virtual AdCoS (based on ADAS models and driver Monitoring Functions so-called MOVIDA - for Monitoring of Visual Distraction and risks Assessment- as presented in D3.6/3.7).

This V-HCD tool chain, based-on COSMODRIVE model, was used to support the MOVIDA-AdCoS virtual design, prototyping and test in WP4, and is one of the "simulation demonstrators" in WP9, specifically dedicated to the implementation and the deployment of the "HF-RTP approach" to automotive domain (see D9.9).

3.5.2 MTT Description

The version of COSMODRIVE model developed for HoliDes is composed of three main modules (Figure 21): a Perception Module (in charge to simulate drivers' perceptive processes, their visual scanning, and their visual distraction status), a Cognition Module (in charge to simulate drivers' situation awareness, anticipation and decision-making processes), and an Action Module (in charge to simulate executive functions and vehicle control abilities) generating driving behaviours. In addition, COSMODRIVE is liable to be monitored by a virtual AdCoS (more particularly by MOVIDA-AdCoS developed by IFS in WP3; see D3.7) and also to adapt its driving behaviour according to warning generated by the AdCoS.

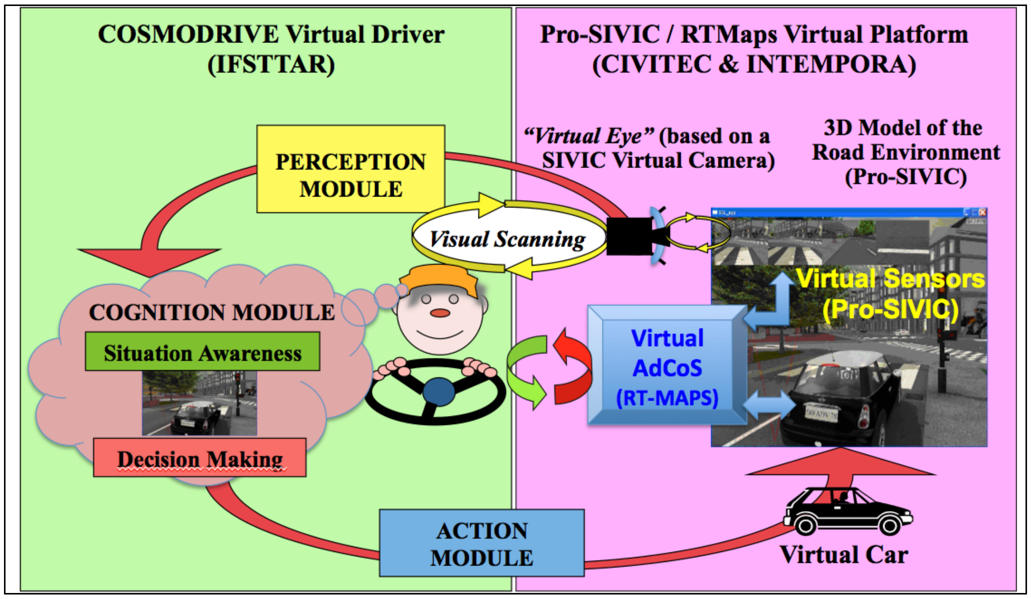

COSMODRIVE Virtual Driver (IFSTTAR)
Pro-SIVIC / RTMaps Virtual Platform (CIVITEC & INTEMPORA)
PERCEPTION MODULE
"Virtual Eye" (based on a SIVIC Virtual Camera)
3D Model of the Road Environment (Pro-SIVIC)
Visual Scanning
COGNITION MODULE
Situation Awareness
Decision Making
Virtual Sensors (Pro-SIVIC)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

Figure 21: COSMODRIVE use in HoliDes for AdCoS and car driving simulation (supported by Pro-SIVIC and RTMAPS 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 driver's errors in terms of misperception of event, erroneous situational awareness, or inadequate behavioural performance, due to visual distractions (resulting of a secondary task performed while driving, for instance).

3.5.3 Current status

In its current status, COSMODRIVE model is able to simulate human drivers' perception (visual scanning and visual distraction status), cognitive processes (like situation awareness, decision-making and action planning) and driving behaviour (i.e. action implementation to pilot the virtual car). This model can be used to simulate a car driver more or less distracted, driving a virtual car equipped or not with the MOVIDA-AdCoS.

3.5.3.1 Simulation of drivers' perception and distraction state

In the frame of HoliDes, one of the core components of COSMODRIVE for MOVIDA-AdCoS design and test is the Virtual Eye of the perception module. This virtual eye includes 3 visual field zones: the central zone corresponding to foveal vision (solid angle of 2.5° centred on the fixation point) with a high visual acuity, para-foveal vision (from 2.5° to 9°), and peripheral vision (from 9° to 150°), allowing only the perception of dynamic events (a more detailed description of COSMODRIVE virtual eye was provided in D2.6).

Moreover, two complementary processes are implemented in the Perception module of COSMODRIVE to simulate the human driver's perceptive functions while driving a car. The first one, named *perceptive integration*, is a "data-driven" process (i.e. bottom-up integration based on a set of perceptive algorithms) and allows the cognitive integration of environmental pieces of information into the Cognition Module, according to their saliencies for the human eye (as illustrated in Figure 22).

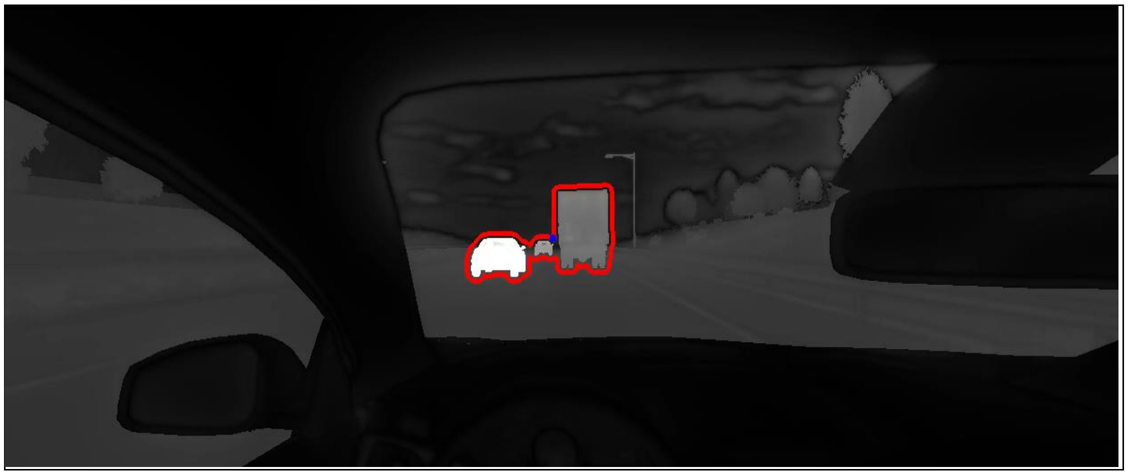


Figure 22: Saliency map implemented by COSMODRIVE perception module

The second process is the *perceptive exploration* (referring to Neisser's theory of perceptive cycle; Bellet et al, 2012) which is a "knowledge-driven" process (i.e. top-down) in charge to continuously update the driver's mental model of the road environment in the Cognition Module, and to dynamically explore the road scene according to driver's situation awareness, risk assessment, intentions and decisions to be made.

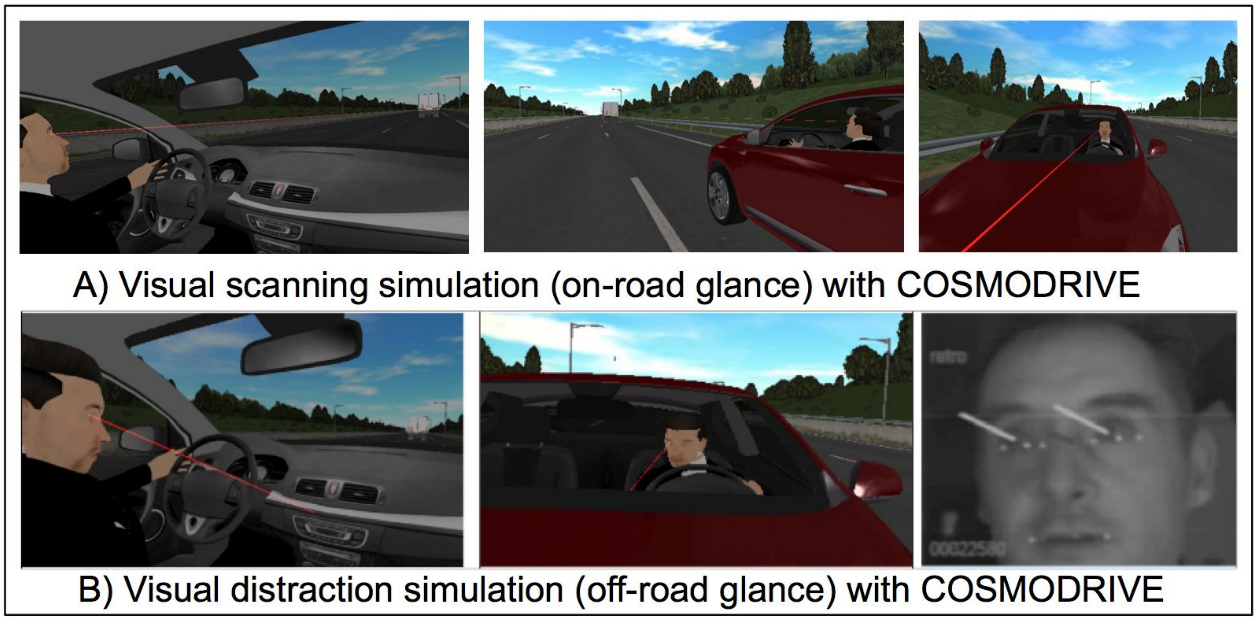

A) Visual scanning simulation (on-road glance) with COSMODRIVE
B) Visual distraction simulation (off-road glance) with COSMODRIVE

Figure 23: Simulation of Drivers' visual Scanning with COSMODRIVE

From this 2nd process, COSMODRIVE is able to dynamically explore the road environment with its virtual eye and thus to simulate real drivers' visual scanning. Visual strategies take the form of a set of *fixation points* which are the "outputs" of COSMODRIVE model to be then monitored by MOVIDA-AdCoS. By observing COMSODRIVE, this AdCoS may analyse drivers' visual scanning and assess their visual distraction status at a given time (like "off-road" glance, for instance). The Figure 23 gives some examples of drivers' visual scanning simulations with COSMODRIVE model, providing similar outputs of data collected with an eye tracking system among real human drivers (as illustrated on the left view).

3.5.3.2 Simulation of drivers' Situation Awareness, Anticipation and Decision-Making processes

Perceptive data collected by the virtual eye and processed by the Perception Module are then integrated in the Cognition Module of COSMODRIVE (Figure 24). The key-component of this Cognition module are "Mental Representations" (Bellet et al., 2009), corresponding to the driver's Situation Awareness according to Endsley's definition of this concept (1995): *the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future*.

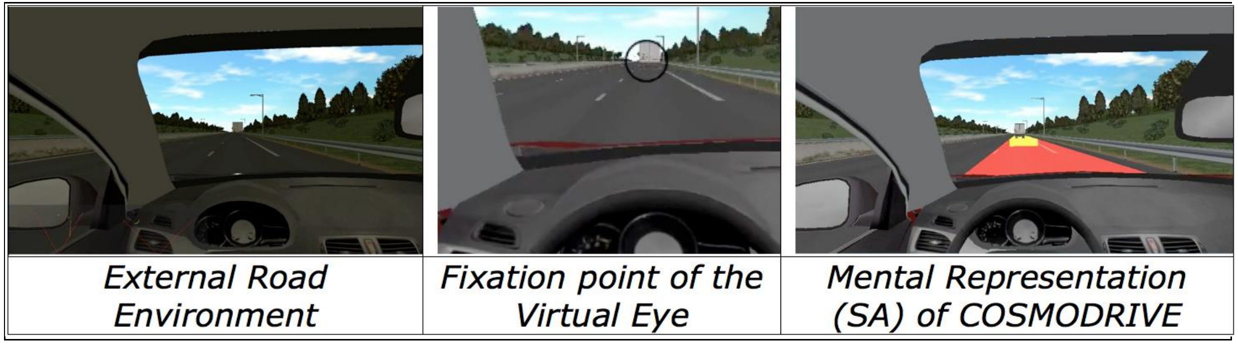

External Road Environment
Fixation point of the Virtual Eye
Mental Representation (SA) of COSMODRIVE

Figure 24: Mental Representation elaboration from Perceptive data

Mental representations, as mental models of the driving situation, are dynamically formulated in working memory through a matching process between (i) information perceived in the external environment and (ii) pre-existing driving knowledge, that are modelling in COSMODRIVE as “Driving Schemas” and “Envelop Zones” (described in D2.6). These mental representations provide ego-centred and a goal-oriented understanding of the traffic situation. They take the form of a Three-Dimensional models (i.e. temporal–spatial) of the road environment, liable to be mentally handled by the driver, in order to support anticipation through cognitive simulations, and thus providing expectations on future situational states. This cognitive process of anticipation is based in COSMODRIVE on a “mental deployment” process (Bellet et al., 2009). The following figure (Figure 25) provides an example of such mental deployment in order to make the decision to implement (or not) an overtaking manoeuvre of a slow truck, by managing the collision risk with the surrounding traffic. To support its decision-making, COMSODRIVE mentally explore alternative driving behaviours (for instance, by immediately implementing a Lane Change manoeuvre with the current speed versus after having braking and reducing the ego-car speed) in order to check their feasibility and to assess their respective level of risk. From the results of these mental deployments, COSMODRIVE make its decision by selecting the less critical behaviour. Then, this planned behaviour at the cognitive level is given to the Action module in order to be effectively implemented by COSMODRIVE (through action on vehicle controls).

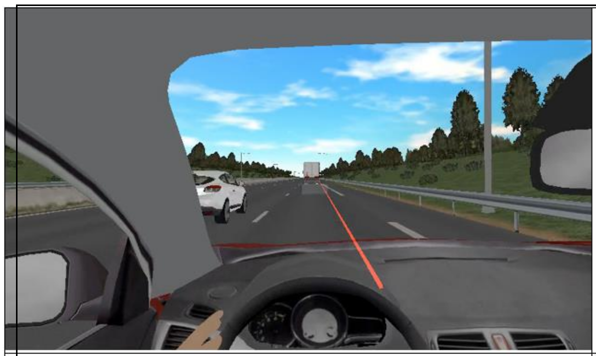
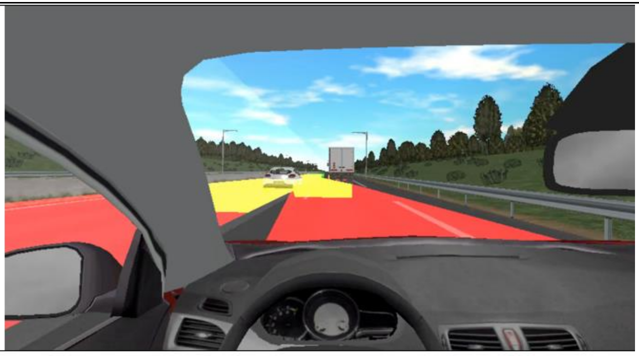
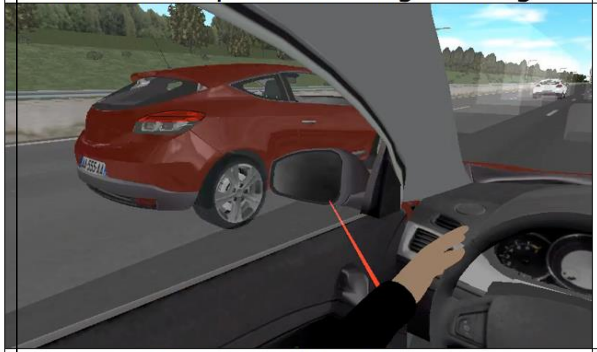
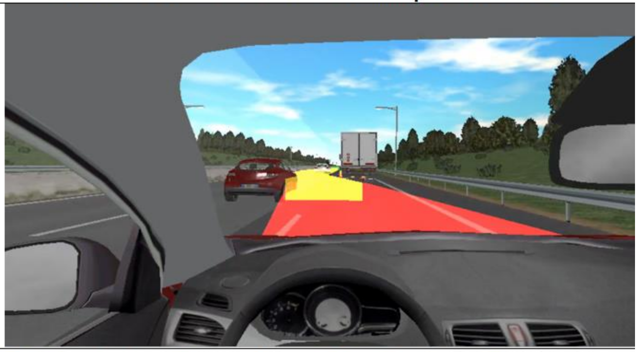
	
Driving Situation: Approaching to a slow truck + Keep the Same Speed for Lange Change	LC Decision based-on *Mental Deployment*: Collision risk with an incoming car (red) is detected => LC is not implemented
	
Driving Situation: Approaching to the slow truck + Braking before Lane Change	LC Decision based-on *Mental Deployment*: Collision risk with the incoming red car is managed => LC implemented after braking

Figure 25: Mental Representation elaboration from Perceptive data

When driving, human drivers continually update their Situational Awareness (SA) as and when they dynamically progress on the road. SA contents depend on the aims the drivers pursue, their short-term intentions (i.e. tactical goals, such as changing of lane for overtaking) and their long-term objectives (i.e. strategic goals, such as reaching their final destination within a given time), the attentional resources they allocated to the driving task and their visual scanning of the road environment. They are the starting point of all the decision-making and behaviours implemented by the driver.

3.5.3.3 Simulation of visual distraction risks with COSMODRIVE

In case of a visual distraction, the mental model updating may be negatively impacted, more particularly in case of too long off-road glance and/or in case of unexpected change in the driving environment. Figure 26 presents a typical example of erroneous Situation Awareness of the driver due to visual distraction, as simulated with COSMODRIVE on the V-HCD platform. In this situation, the followed truck brakes when COSMODRIVE is visually distracted (fixation point of the virtual eye on the car radio; central view). Consequently, the mental model of the driver (left view) is not correctly updated regarding this unexpected situational change (right view).

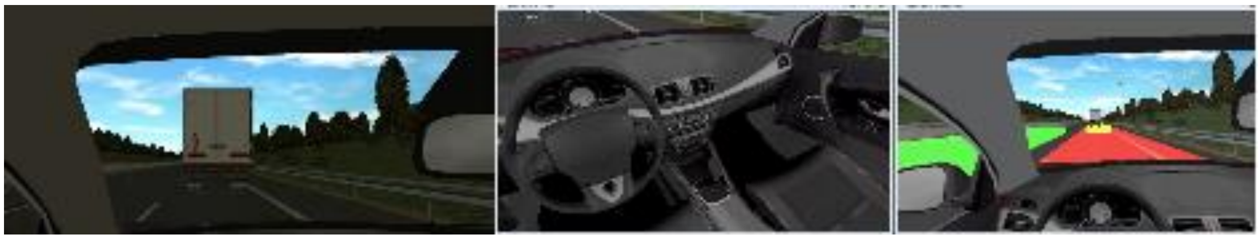


Figure 26: Erroneous updating of Driver's SA due to visual distraction

3.5.4 Integration status

COSMODRIVE integration as an "HF component" of a tailored HF-RTP for automotive domain was a joint effort of IFSTTAR, INTEMPORA and CIVITEC, implemented in WP4 and WP9 (complementary descriptions of this integrative work are also presented in D4.7 and D9.7). To support the virtual design, prototyping and test of the MOVIDA-AdCoS (described in D3.7), a Virtual Human Centred Design platform (so-called V-HCD) has been jointly developed by IFS, CVT and INT, as an example of a tailored HF-RTP based on RTMaps software specifically dedicated to dynamic simulations of virtual AdCoS (see detailed description in D4.7). This V-HCD integrative platform was completed during HoliDes and then used in WP4 to virtual design and evaluate the MOVIDA-ADCoS and to provide one of the simulation Demonstrators in WP9 (see D.9.9).

Synthetically, the V-HCD integrates 4 main MTTs: (1) COSMODRIVE model able to visually explore the road environment from a "virtual eye" and to drive (2) a virtual car (simulated with Pro-SIVIC) (3) equipped with virtual ADAS (Advanced Driving Aid Systems) and with the MOVIDA-AdCoS (simulated with RTMaps and Pro-SIVIC), for dynamically progressing in (4) a virtual 3-D road environment (simulated with Pro-SIVIC). To support the HF-based design process in HoliDes, the V-HCD platform is used to generate dynamic simulations of car sensors, ADAS and AdCoS when interacting with

COSMODRIVE driver model (i.e. the “HF component” of the RTP), in order to support the virtual prototyping and test of MOVIDA.

All the MTTs required for the MOVIDA-AdCoS and its design process with the V-HCD platform have been integrated into the RTMaps software. The RTMaps diagram presented in Figure 27 provides an overview of the COSMODRIVE and MOVIDA integration/interfacing with this software. On this figure, the MOVIDA-AdCoS sub-diagram receives inputs (1) from COSMODRIVE regarding both drivers’ visual behavior (to assess the visual distraction state of the driver) and their actions on vehicle commands (for lateral and longitudinal control of a Pro-SIVIC ego-car) and (2) from the ADAS virtually simulated with Pro-SIVIC and RTMaps. In addition, MOVIDA-AdCoS generates outputs towards the Pro-SIVIC virtual car to implement COSMODRIVE and/or MOVIDA-AdCoS driving actions.

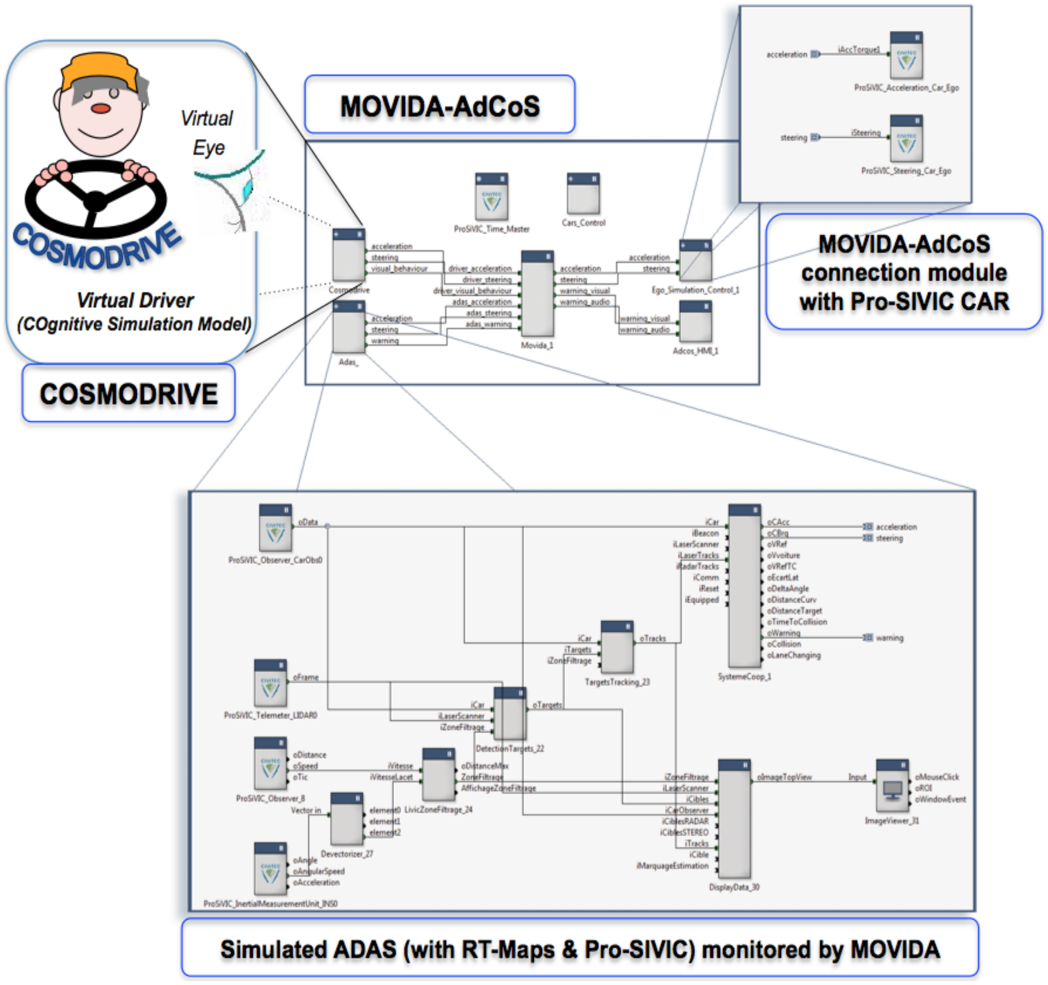

MOVIDA-AdCoS
Virtual Eye
COSMODRIVE
Virtual Driver (COgnitive Simulation Model)
COSMODRIVE
MOVIDA-AdCoS connection module with Pro-SIVIC CAR
Simulated ADAS (with RT-Maps & Pro-SIVIC) monitored by MOVIDA

Figure 27: RTMaps diagram for MOVIDA-AdCoS tests with COSMODRIVE

In WP4 and WP9, the objective was to use COSMODRIVE-based simulations to support MOVIDA-AdCoS Validation and Verification from dynamic simulations, by considering the future use of this AdCoS by human drivers (i.e., end-users, as simulated with COSMODRIVE).

Figure 28 provides an illustration of V-HCD use for evaluating the advantage a simulated MOVIDA-AdCoS regarding road safety. At phase 1, the COSMODRIVE is visually distracted (i.e. the Virtual eye is focused on the dashboard). At this time, the driver’s mental model (i.e. Situation Awareness) of the road environment is correct because the off-road glance is only starting.

At phase 2, a white car is overtaking the driver, and the followed truck starts to brake. Due to the visual distraction, the Situation Awareness of COSMODRIVE is not updated (i.e. the truck is closer and the white car is present in the reality (as presented on the left image), versus the truck is still far and there is no white car in the driver's Situational Awareness (as presented on the right image)). In parallel, the MOVIDA-AdCoS computes (from the simulated radars) the Inter-Vehicular Time (IVT) with the lead truck and also detects an approaching car in the blind spot area (on left lane). MOVIDA functions also diagnose the visual distraction status of COSMODRIVE, and generate warnings (visual and auditory) to alert the driver of the different collision risks (with the truck and in addition with the lateral car in case of lane change).

At phase 3, the warned COSMODRIVE focuses its virtual eye on the road environment and updates its situational awareness. At this time, the Driver/COSMODRIVE is aware of the risk and able to manage it.

At phase 4, the incoming red car overtakes the ego car, and the MOVIDA-AdCoS informs the driver that a Lane Change is now possible.

Integrated simulations based on "COSMODRIVE + MOVIDA-AdCoS + ADAS + Car Sensors" were used during the virtual prototyping process of the MOVIDA-AdCoS (implemented in WP4; see D4.7) in order to progressively evaluate its *efficiency* (i.e. how well this driving aid works?) and its *effectiveness* according to real human driver's needs (i.e. how useful this driving aid is?). During this design cycle, I-Deep functionalities of RTMaps were used to replay several times driving scenarios (like the scenario presented in Figure 28 for instance) and testing different timings of the AdCoS warnings with the aim to identify, for instance, the last moment when human drivers are able to manage the risk from these warning, and when they are not able to avoid the collision even if warned, requiring in this case to activate an automatic vehicle control taking by the MOVIDA-AdCoS (e.g. emergency braking for collision avoidance or for collision mitigation).

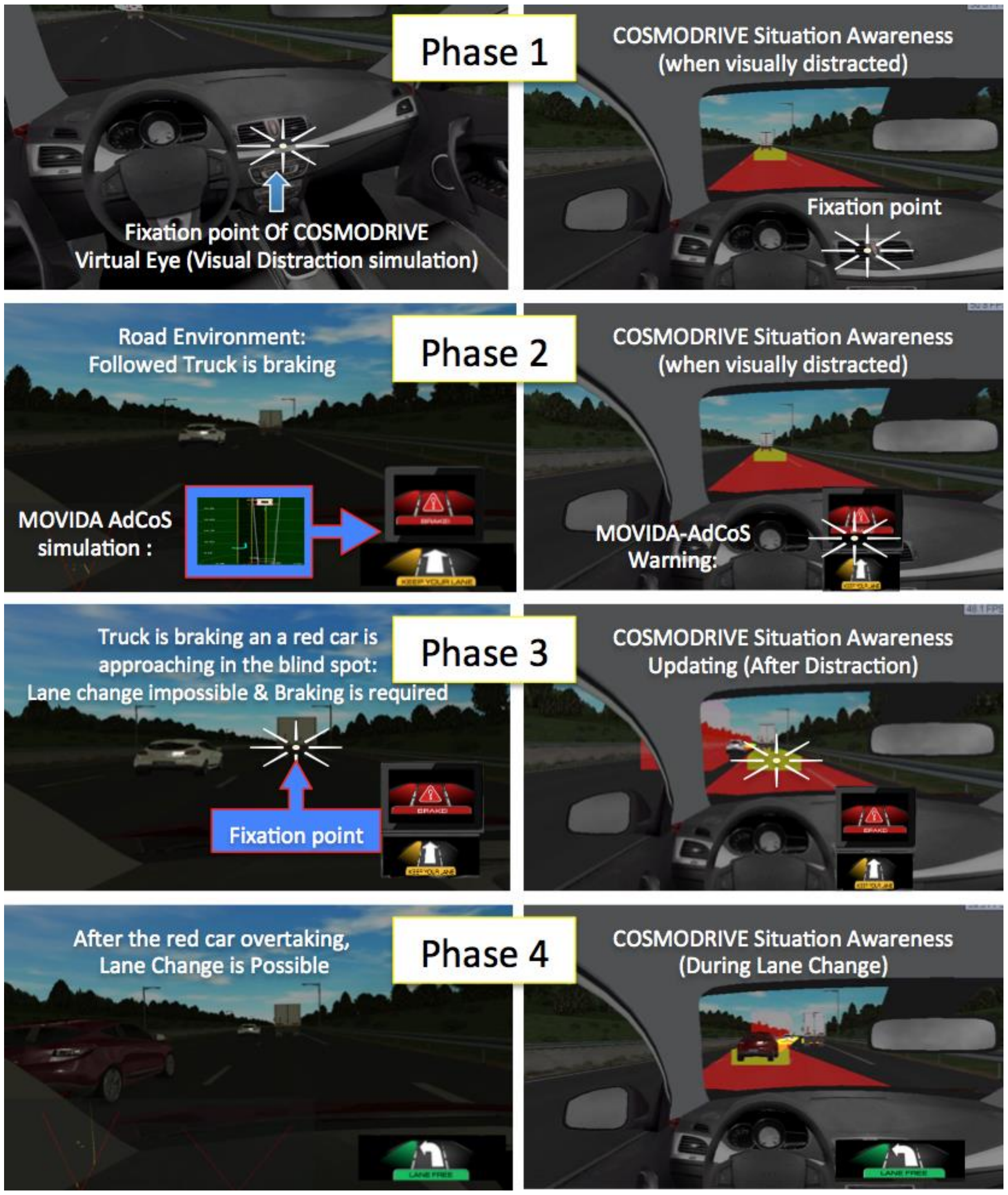

Phase 1
Fixation point Of COSMODRIVE Virtual Eye (Visual Distraction simulation)
COSMODRIVE Situation Awareness (when visually distracted)
Fixation point
Phase 2
Road Environment: Followed Truck is braking
MOVIDA AdCoS simulation :
COSMODRIVE Situation Awareness (when visually distracted)
MOVIDA-AdCoS Warning:
Phase 3
Truck is braking an a red car is approaching in the blind spot: Lane change impossible & Braking is required
Fixation point
COSMODRIVE Situation Awareness Updating (After Distraction)
Phase 4
After the red car overtaking, Lane Change is Possible
COSMODRIVE Situation Awareness (During Lane Change)

Figure 28: COSMODRIVE-Based simulation to support AdCoS virtual design

3.6 Great SPN for MDPN (UTO)

3.6.1 State of the art

GreatSPN is a tool developed by the University of Torino in the last 30 years. It is a software framework for the verification of systems, represented with the Petri net formalism, that has been used with success to model many real cases, like bandwidth load in multiprocessor systems, chemical reaction networks, peer-to-peer systems, UML diagrams, and other. The extension to include Markov Decision Process (MDP) solvers and Markov Decision Petri Nets (MDPN) as a Petri net language for the high level definition of MDP is instead work that started a few years back and is it still under development, in particular to adapt it to the needs of HoliDes. Adaptation concerns the graphical user interface (GUI) and the MDPN/BMDP[7] solver, as described throughout this section.
The GUI allows drawing the models graphically, using the Petri net formalism. The interface of the GUI is shown in Figure 29.

[7] BMDP means Bounded-parameter Markov Decision Process (see next paragraphs for more details).

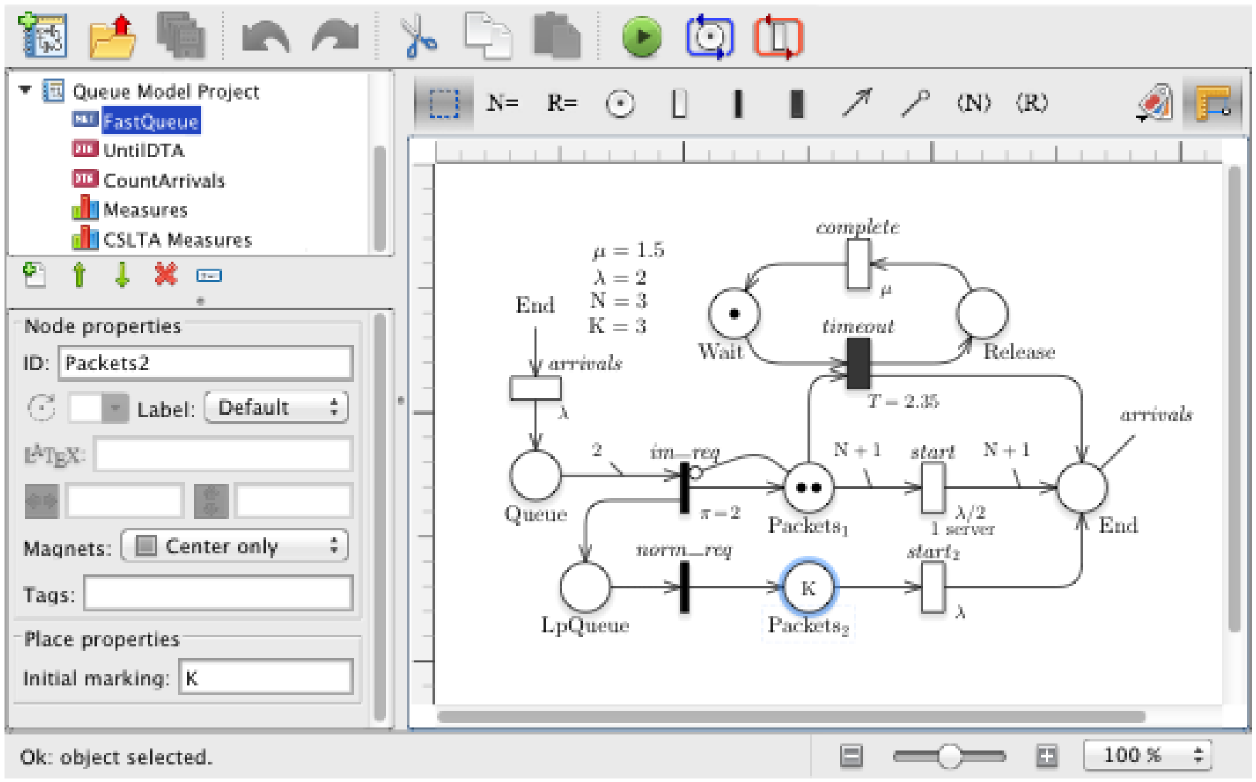
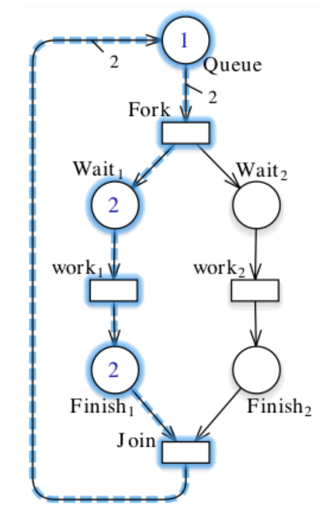
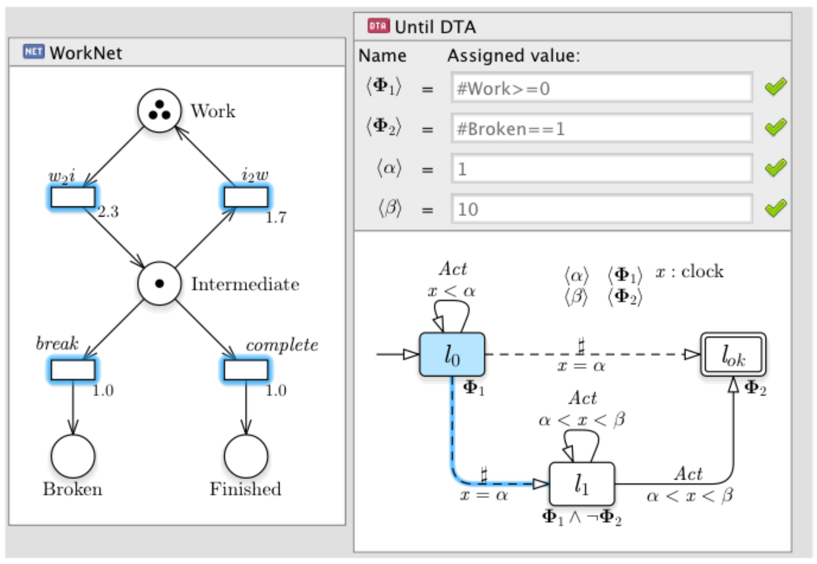


Figure 29: The GreatSPN graphical interface

The general data model of the GreatSPN editor is a compositional model where each component is a 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, WYSIWYG 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 30.



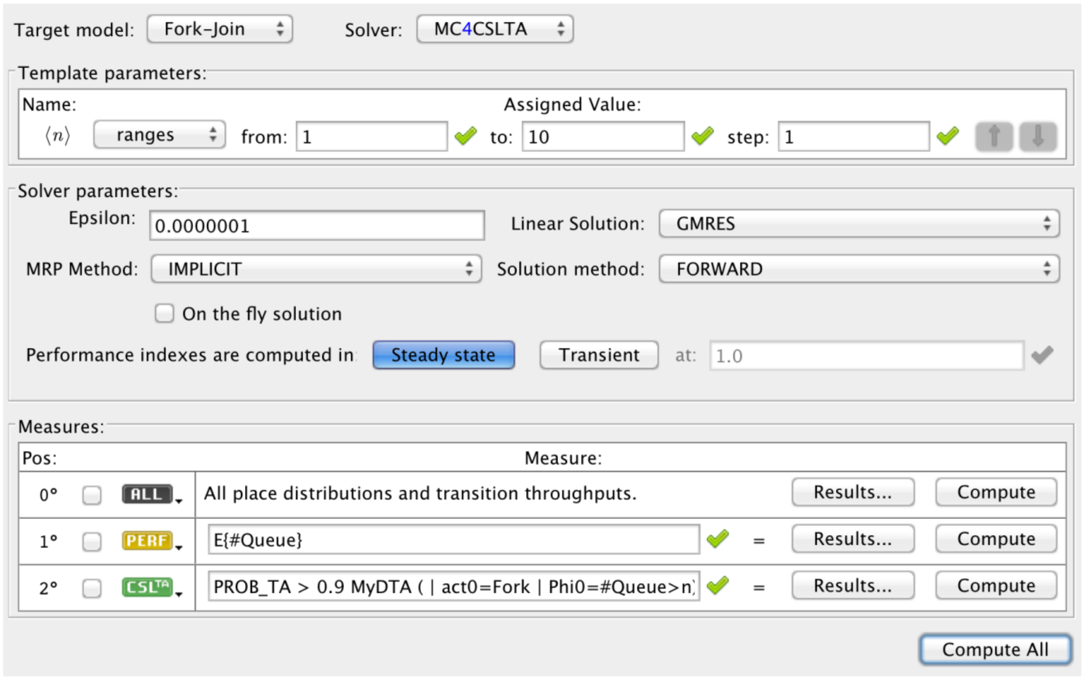
(**A**) Interactive visualization of a P-semiflow of a GSPN.


WorkNet
Work
w_2i
i_2w
2.3
1.7
Intermediate
break
complete
1.0
1.0
Broken
Finished
Until DTA
Name
Assigned value:
$\langle\Phi_1\rangle$ = #Work>=0
$\langle\Phi_2\rangle$ = #Broken==1
$\langle\alpha\rangle$ = 1
$\langle\beta\rangle$ = 10
Act
$x < \alpha$
$\langle\alpha\rangle$ $\langle\Phi_1\rangle$ x : clock
$\langle\beta\rangle$ $\langle\Phi_2\rangle$
l_0
$\sharp$
$x = \alpha$
l_{ok}
Φ_1
Φ_2
Act
$\alpha < x < \beta$
$\sharp$
$x = \alpha$
l_1
Act
$\alpha < x < \beta$
$\Phi_1 \wedge \neg\Phi_2$

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

Figure 30: An example of interactive testing in GreatSPN

Invariant visualization supports P-semiflows and T-semiflows, which characterize the behaviour of the model (A), while interactive simulation (B) allows the user to play with the model, activating its transitions to simulate the behaviour of the system and observe the result.
Once a model has been drawn, performance indices can be computed on it using a collection of numerical solvers. A batch of indices can be specified through the GUI, which invokes the solvers, performs the computation and shows the results interactively. Figure 31 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 31: 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 32, which displays an MDPN model drawn with the GreatSPN GUI.

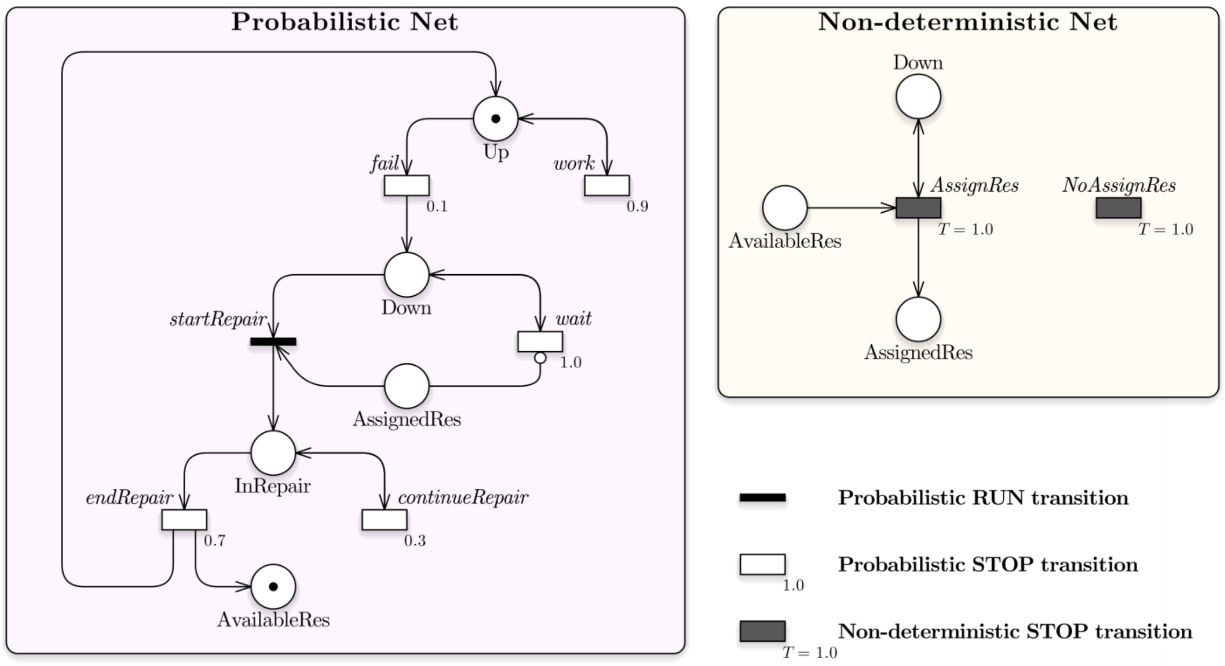


Figure 32: An MDPN in GreatSPN

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.

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 consists 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 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 also subsume MDPs, but also permit both exponentially timed and immediate transitions. Unfortunately, a tool does not support them.
A number of process algebras featuring nondeterministic and probabilistic choice have been introduced; reader can refer to [Probabilistic extensions of process algebras] for an overview of a number of these.

3.6.2 Current status

The GreatSPN framework provides a modular system to design and solve MDPN models by means of specific modules. The current status of the framework has been extended to incorporate uncertainty on model parameters, to account for real-data coming from sensors, as needed by the AdCoS development.

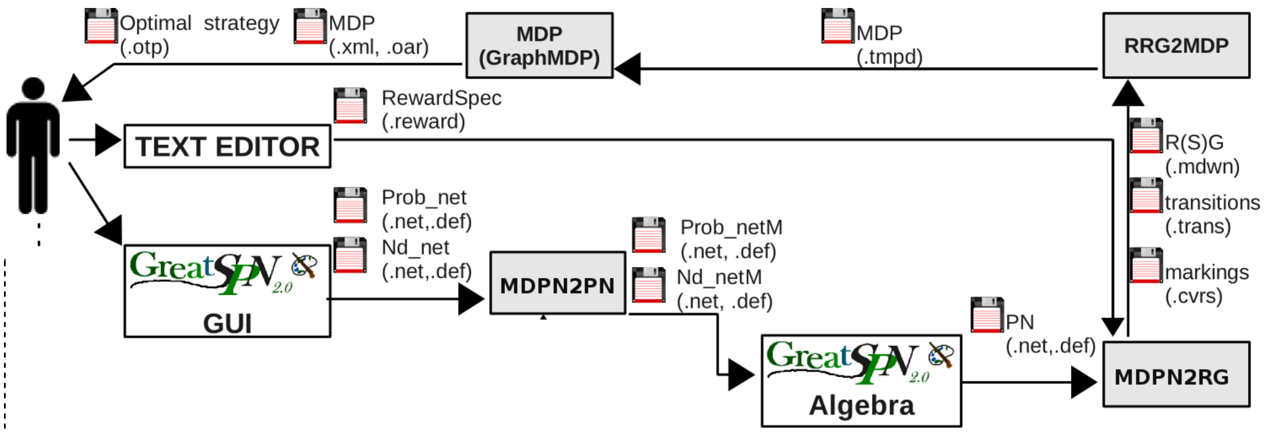


Figure 33: MDPN solver Architecture

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. To account for the uncertainty, the MDPN model formalism has been extended: parameters may now be expressed using intervals instead of exact data, and the underlying process becomes a Bounded-parameter Markov Decision Process (BMDP).

The architecture of this MDPN framework is depicted in Figure 33. The user must specify PN^{nd} and PN^{pr} subnets (in Figure 32 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 BMDP 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. Rewards may also be characterized by uncertainty, like the other model parameters.

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 BMDP actions) and markings description (the BMDP states), needed to compute the value of the reward function associated with the BMDP states and actions; (4) module ***RG2MDP,*** generates the final BMDP: the states of the BMDP correspond to the *tangible* states produced by the previous module, the BMDP 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.
A new solver has been developed for BMDP models. In order to make the BMDP solution 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 BMDP file is produced in an efficient format which is accepted in input by the ***BMDP*** solver module (based on the *APNN toolbox* library), that produces the optimal strategy and corresponding optimal reward value.

3.6.3 Integration status

The integration of the MDP solver of GreatSPN into the HF-RTP platform was done in the form of a RTMaps component. The component is designed to take as input a set of asynchronous data flows from multiple physical sensors and data analysers, and produce as output the *MDP strategy* and the estimated *warning level*, that realize the CO-PILOT logic.

Figure 34 shows the structure of the RTMaps component that contains the MDP solver of GreatSPN.

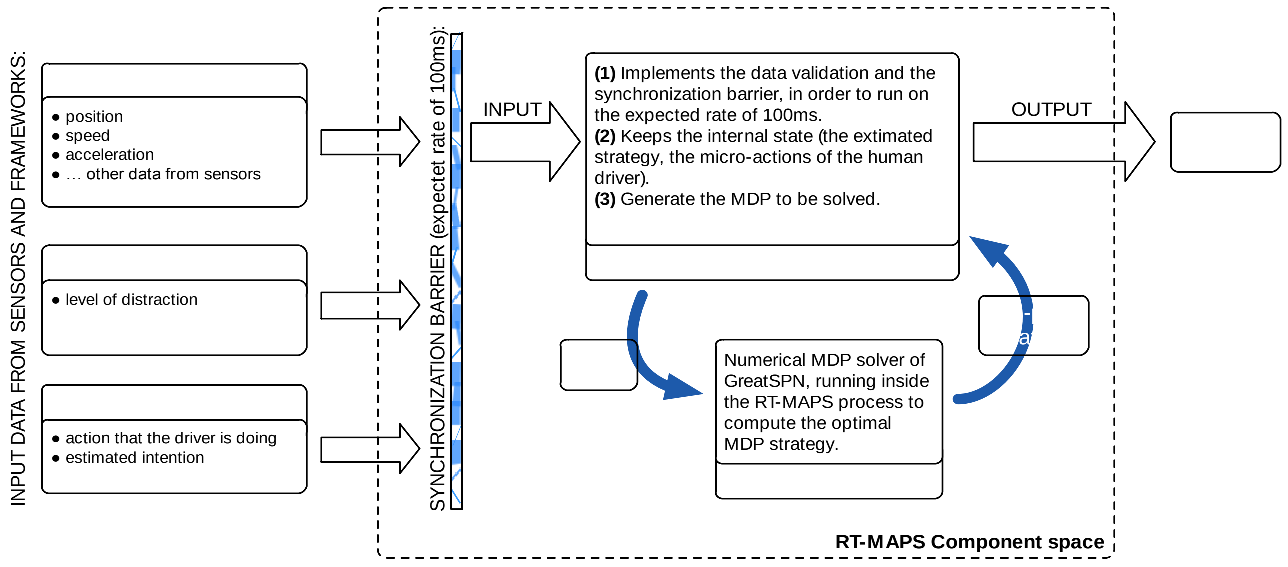

INPUT DATA FROM SENSORS AND FRAMEWORKS:
• position
• speed
• acceleration
• … other data from sensors
• level of distraction
• action that the driver is doing
• estimated intention
SYNCHRONIZATION BARRIER (expectet rate of 100ms):
INPUT
(1) Implements the data validation and the synchronization barrier, 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.
OUTPUT
Numerical MDP solver of GreatSPN, running inside the RT-MAPS process to compute the optimal MDP strategy.
RT-MAPS Component space

Figure 34: integration of MDP in RTMaps

The component takes as input a certain amount of data, provided by the sensors, the prediction of intention module and the distraction classifiers, that are used to generate the solution MDP. However, since data from other components and sensors are not synchronized (each component/sensor works at its functional rate), a synchronization of this information is required. Synchronization can be done directly using the RTMaps 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 as 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 Driver Intention Recognition (DIR) Models (former BAD MoB) (OFF)

3.7.1 State of the art

The modelling of human driving behaviour has been an extensive area of research in the domain of transportation systems and has been conducted since at least Gibson and Crook's Field of Safe Travel Model (Gibson and Crooks, 1938). Despite that, no all-purpose, generic, comprehensive, and verifiable model of human driver behaviour has been found yet (Ranney 1994). Different models emphasize different and sometimes very specific aspects of driving behaviour, like accident causation, training, behavioural adaptation, situation awareness, etc., and are used for many different purposes, including micro- and macroscopic simulation of driver behaviour, autonomous control, safety analysis resp. risk assessment, and manoeuvre resp. intention recognition. As such, "the variety of models of the driving task is almost as numerous as the number of authors who have contributed to the models" (Carsten 2007, p. 105), which renders any sufficient state of the art of driver modelling impossible. Due to the utilization of the BAD MoB models as a means of intention recognition in WP9, we will therefore focus on the state of the art of *driver intention recognition*.

Michon (1985) provided an influential conceptual model for the human driving task, describing it as a hierarchical structured task with three levels of skills and control: strategical (planning), tactical (manoeuvring), and operational (control). At the strategic level, the general planning of a journey is handled, e.g., the driver chooses the route and evaluates resulting costs and time consumption. At the tactical level, the driver has to select manoeuvres, e.g., turning at an intersection or initiating a lane change manoeuvre. At the control level, the driver has to execute simple (and for experienced drivers mostly autonomous) action patterns, which together form a manoeuvre or behaviour. Examples are braking manoeuvres in order to keep a safe distance to a leading vehicle or turning the wheel to perform a lane-change. That said, driver intention recognition usually focusses on the recognition of *tactical manoeuvre intentions*, like e.g., lane-changes or overtaking manoeuvres.

Models for driver intention recognition can be distinguished by the underlying modelling technique and the input used for intention estimation. Concerning the modelling techniques, the state of the art in intention, resp. manoeuvre recognition is mostly based on Dynamic Bayesian Networks (DBNs), most prominently Hidden Markov Models (HMMs) and their variants, or probabilistic and non-probabilistic discriminative models. A comparative review of works on manoeuvre intention estimation can be found in Lefèvre et al. (2014b) and Doshi and Trivedi (2011). An additional overview can be found e.g., in Kobiela (2011) and Börger (2013).

For intention recognition based on DBNs, in general, the basic idea can be summarized as follows: For each addressed manoeuvre, a distinct DBN is learnt that models the dynamic evolution of the vehicle state and/or position for the specific manoeuvre. Given a new sequence of observations, the actual manoeuvre intention is then estimated by comparing the likelihood of observations for each DBN (e.g., Oliver and Pentland, 2000, Torkkola et al. 2005, Kumagai, 2006, Tay, 2009, and Lieber et al., 2012). To provide an example, Oliver and Pentland (2000) used seven distinct HMMs to recognize each of seven driving manoeuvres, evaluating four different combinations of feature vectors consisting of vehicle data, lane position information, and driver gaze information. On average, the resulting models were able to recognize the addressed manoeuvres one second before “any significant (20% deviation) change in the car or contextual signals” took place (Oliver and Pentland, 2000).

For intention recognition based on discriminative (probabilistic) models, commonly used techniques are (non-probabilistic) Support Vector Machines (SVMs) (e.g., Aoude et al., 2010, Kumar et al., 2013, and Mandalia and Salvucci, 2005), Multi-Layer Perceptrons (MLPs) (Klingelschmitt et al., 2014), or logistic regressions (Garzia-Ortiz, 2010). To the best of our knowledge, the most sophisticated model up to date is the discriminative model described by Doshi et al. (2011) resp. Morris et al. (2011). They used Relevance Vector Machines (RVMs) for learning a model for online recognition of lane-change intentions, which can be seen as a Bayesian alternative to SVMs, in that they provide a probabilistic classification. According to Doshi et al. (2011), several advantages of this methodology motivate the use of RVMs over other algorithms, such as SVMs and HMMs. The RVM can sift through large feature sets and obtain a sparse data representation, which is especially useful in identifying a small set of useful features for intention recognition. Multimodal data from various sets of sensors can thereby be combined easily, with the RVM automatically choosing discriminating cues from each modality. The resulting sparse representation allows for quick computation and classification in real time and real-world conditions with limited hardware.

For driver intention recognition based on DBNs, the current underlying methodology has the severe disadvantage that a manoeuvre has to be initiated in order to be distinguishable from other hypothetical manoeuvres. In general, discriminative approaches don't suffer from this weakness, which explains their popularity for intention recognition. However, discriminative models are not easily interpretable and in general cannot handle missing input, i.e., they are restricted to recognition rather than prediction.

Concerning the input used for driver intention recognition, with the notable exception of Doshi et al. (2011), resp. Morris et al. (2011) and Ohn-Bar et al. (2014), most proposed models limit the features used for intention recognition to information about the driver (e.g., gaze directions), the vehicle state (e.g., speed and acceleration, as determined by the driver's behaviour) and (seldom) potentially course information, while neglecting potential vehicles in the vicinity of the driver (Levèvre et al., 2014b). As a consequence, the proposed models are unable to give suggestions solely based on the external context and require that the vehicle is actually controlled by the human driver.

Models for intention recognition in overtaking resp. lane-change scenarios are usually compared in respect to their *prediction horizon*, i.e. the time span between the recognition of an intention and the actual lane-crossing. Current models for intention prediction are sufficient to predict single specific behaviours of the human driver up to approx. three seconds (e.g., Garcia-Ortiz et al., 2011, Morris et al., 2011, Doshi and Trivedi, 2008, and Lefèvre et al., 2014a). To give some recent examples, Ohn-Bar et al. (2014) used discriminative Latent-Dynamic Conditional Random Fields based on sensor information of vehicle dynamics, vehicles in the vicinity of the driver and course information, as well as sensor information of the driver to recognize overtaking manoeuvres. They found very good results in recognizing overtaking manoeuvres up to two seconds prior to the actual lane crossing. Bi et al (2015) proposed the use of Queuing Network-Based Driver Models, a control-theoretic approach that uses a queuing network system based on neuroscience and psychological findings. Similar to approaches based on DBNs, they use the driver models to predict a set of five template manoeuvres (normal and emergency lane changes to the left and right lane, and lane-keeping) and then calculate the root-mean-square error (RMSE) between the actual steering angle sequence and the simulated templates. The most probable intention is then given by the template that minimizes the RMSE. From simulator studies, they report detections within 0.325 to 0.268s of the steering manoeuvre onset, which from a visual inspection equates to a prediction horizon of approx. 3.0-3.5s.

3.7.2 MTT Description

Bayesian Autonomous Driver Mixture-of-Behaviours (BAD MoB) models are a technique for modelling the human driver behavior based on Dynamic Bayesian Networks (DBNs). As such, they can be seen as probabilistic human operator models. In the context of HoliDes, BAD MoB models are used for driver intention recognition as a means of context assessment.

As depicted in Deliverable D2.6 "Modelling Techniques and Tools Vs1.8", we used simulator data to investigate both a *discriminative* and a *generative* approach for the use of BAD MoB models for driver intention recognition. Early prototypes suggested that a generative approach is more suited for the intended use, in contrast to the "classical" discriminative approach of BAD MoB models, so we focussed the further development on the generative approach. To reduce the risk of confusion and to prevent a dilution of these different approaches, we rename the resulting models for driver intention recognition "*Driver Intention Recognition*" (DIR) models. Within HoliDes, the DIR models are used within the MTT developed in WP3 and utilized in the WP9 Automotive AdCoS "Adapted Assistance", called the Driver Intention Recognition (DIR) *module* to provide the AdCoS application with prediction about the intentions of human drivers.

3.7.3 Current status

Between July and November 2015, CRF conducted a driving study with 44 participants on Italian highways near Turin to collect data for the development and evaluation of different non-lifecycle MTT composing the AdCoS "Adapted Assistance". The participants entered the two-lane highway

"A55 Torino-Pinerolo" at the "SP142" entry Orbassano and exited the "A55 Torino-Pinerolo" after approx. 15 km at the "Riva di Pinerolo" exit to return the same highway back to Turin. After the original entry they changed to the three-lane highway from Orbassano to Moncalieri for approx. 50km, exited and returned to the CRF location. The participants were instructed to follow the obligation to drive on the right lane on the two-lane highway, but were free to choose between the right and the middle lane on the three-lane highway. Once chosen, they were instructed to avoid lane changes unless necessary. At random moments but well in advance, a CRF employee in the vehicle instructed the participants to perform an overtaking maneuver once they felt comfortable and safe, using the indicator during the process. In this manner, approx. 20 overtaking maneuvers per trial were collected. The participants were allowed to overtake multiple vehicles at once but were instructed to not perform overtaking maneuvers involving more than two lanes.

Of the resulting 44 trials, 30 trials, obtained at the end of July and during September, were suitable for the development and evaluation of DIR models. As the target scenario for the AdCoS "Adapted Assistance" and the DIR module focusses on two-lane highways and as file size limits rendered the sensor information inaccessible after approx. 15 min. of driving, we focused on the data of the first section of each trial, beginning with entering the "A55 Torino-Pinerolo" at the "SP142" entry and ending when exiting the "A55 Torino-Pinerolo" at the "Riva di Pinerolo" exit.

From the available 30 trials, six were discarded for the development of the DIR models due to insufficient data quality or out-of-sync errors during data collection. For each of the remaining 24 trials, we used the data pre-processing component of the DIR module in RTMaps to built up a database of experimental data used as input for learning the DIR models. We manually annotated each sample of the experimental data with whether the driver was performing a *lane change to the fast lane*, a *lane change to the slow lane*, or just *lane-keeping* driving behaviour. After this manual annotation, we automatically annotated each data sample with whether the driver intended to drive on the fast or on the slow lane, assuming that a change in the target lane intention was present up to one second prior to the annotated beginning of a lane change manoeuvre. From this annotated experimental data, we randomly selected 17 trials as training data (169294 samples or approx. 141

min of driving) and reserved seven trials for testing and evaluation purposes (69953 samples or approx. 58 min of driving).

Given the training data, we used machine-learning methods to learn a DIR model to be utilized in the DIR module. In the following, let

- L denote a binary random variable with the possible values $\mathrm{Val}(L) = \{\mathrm{slow\_lane}, \mathrm{fast\_lane}\}$, representing context information in terms of the lane, the driver is currently inhabiting,
- I denote a binary random variable with the possible values $\mathrm{Val}(I) = \{\mathrm{slow\_lane\_intention}, \mathrm{fast\_lane\_intention}\}$ that represents the behavioural intentions of a driver in respect to the lane he/she intends wants to inhabit,
- B denote a discrete random variable with the possible values $\mathrm{Val}(B) = \{\text{lane change left, lane change right, lane-following}\}$, representing a set of three potential behaviours/manoeuvres,
- A denote a continuous random variable representing the position of a combined acceleration-braking pedal,
- S denote a continuous random variable representing the steering wheel angle,
- and $\boldsymbol{P}$ denote a set of discrete and continuous variables $\boldsymbol{P} = \{P_1, \dots, P_n\}$, representing a selection of perceptual features that are hypothetically available and important for driver intention recognition. In total, we considered 79 perceptual features to be potentially included in the model:
    - In: Discrete variable representing the indicator signal.
    - S: Continuous variable representing the velocity of the ego-vehicle
    - SL: Discrete variable representing the speed limit,
    - SP: Continuous variable representing the difference between the speed limit and the velocity of the ego-vehicle, which we call the speed potential.
    - LP: Continuous variable representing the lateral position of the ego-vehicle in respect to the lane edge of the fast lane.
    - Y: Continuous variable representing the yaw (or heading) angle of the ego-vehicle.
    - ΔY: Continuous variable representing the rate of change of the yaw angle of the ego-vehicle, i.e. the yaw-rate.

For each alter-vehicle of the twelve potential vehicles in the vicinity of the ego-vehicle, we considered the following perceptual features, where X denotes a vehicle classifier (e.g., the lead vehicle):

- E_X: Binary variable representing the existence of a vehicle X in the vicinity of the ego-vehicle
- A_X: Binary variable, representing the area (within a distance of 25m in respect to the ego-vehicle) a vehicle X is inhabiting
- iT_X: Continuous variable representing the inverse time to collision to a vehicle X. If the ego-vehicle follows the vehicle X, the inverse time to collision is defined from the ego-vehicle to the vehicle X, and vice versa otherwise.
- SD_X: Continuous variable representing the speed difference between the ego-vehicle and a vehicle X
- D_X: Continuous variable representing the distance headway to a vehicle X. If the ego-vehicle follows the vehicle X, the distance headway is defined from the ego-vehicle to the vehicle X, and vice versa otherwise
- THW_X: Continuous variable representing the time headway to a vehicle X. If the ego-vehicle follows the vehicle X, the time headway is defined from the ego-vehicle to the vehicle X, and vice versa otherwise

We focussed on a generative modelling approach, where the underlying probabilistic model is based on the assumptions of first-order Markov and time invariance, so that the joint probability density $p(L^{1:T}, I^{1:T}, B^{1:T}, A^{1:T}, S^{1:T}, \boldsymbol{P}^{1:T})$ can be factorized as:

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

As (1) the quality of the training data was not sufficient to reliably learn models for predicting the control-behaviour for lateral and longitudinal control, and (2) such output was not planned to be used within the AdCoS "Adapted Assistance", we focussed on the intention and behaviour recognition aspects and provided the control inputs of the driver as additional input features ($\boldsymbol{P}_* = \boldsymbol{P} \cup \{A, S\}$), resulting in the following factorization:

$$p\left(L^{1:T}, I^{1:T}, B^{1:T}, \boldsymbol{P}_*^{1:T}\right)$$
$$= \prod_{t=1}^{T} p(L^t)\, p\left(I^t, B^t, \boldsymbol{P}_*{}^t \mid I^{t-1}, B^{t-1}, L^t\right)$$

We additionally assumed that the further factorization $p\left(I^t, B^t, \boldsymbol{P}_*{}^t \mid I^{t-1}, B^{t-1}, L^t\right)$ may be described in terms of a (factorized) dynamic model $P(I^t, B^t \mid I^{t-1}, B^{t-1}, L^t)$ and an observation model $p\left(\boldsymbol{P}_*{}^t \mid I^t, B^t, L^t\right)$ and that for each $l^t \in Val(L)$, the exact factorization of $p\left(\boldsymbol{P}_*{}^t \mid I^t, B^t, l^t\right)$ may differ (i.e., we assume the existence of context-specific independencies). As such, the resulting structure can be understood as a factorized *Hidden Markov Model* where the observation model $p(\boldsymbol{P}^t \mid I^t, B^t, L^t)$ is a so-called *Bayesian Multinet* (see e.g., Koller and Friedman, 2009).

Based on the learning procedure already described in D2.6, the learning task can be understood as *feature selection*, in that we try to find a suitable subset of $\boldsymbol{P}_*$ important for recognizing intentions and behaviours by learning a corresponding graph-structure factorizing $p(I^t, B^t, \boldsymbol{P}^t \mid I^{t-1}, B^{t-1}, L^t = \text{slow\_lane})$ and $p(I^t, B^t, \boldsymbol{P}^t \mid I^{t-1}, B^{t-1}, L^t = \text{fast\_lane})$. For learning, we assumed that the observation models $p\left(\boldsymbol{P}_*{}^t \mid I^t, B^t, L^t\right)$ can be factorized in terms of a naïve Bayesian classifier, where we allow certain variables to be conditioned by additional variables describing the context. Without a loss of generality, let $\boldsymbol{P}_{Dep} = \{P_1, \dots, P_n\} \subseteq \boldsymbol{P}_*$ denote a set of perceptual features conditioned by I^t and/or B^t in an observation model $p\left(\boldsymbol{P}_*{}^t \mid I^t, B^t, l^t\right)$, $\boldsymbol{P}_{Ind} = \{P_{n+1}, \dots, P_m\} \subseteq \boldsymbol{P}_*$ denote a set of not conditioned perceptual and $\boldsymbol{Pa}(P_i)$ denote a set of additional parent variables, we assume that an observation model $p\left(\boldsymbol{P}_*{}^t \mid I^t, B^t, l^t\right)$ can be factorized as:

$$p\left(\boldsymbol{P}_*{}^t \mid I^t, B^t, l^t\right) = \prod_{P_i \in \boldsymbol{P}_{Dep}} p\left(P_i^t \mid I^t, B^t, \boldsymbol{Pa}(P_i)\right) \prod_{P_j \in \boldsymbol{P}_{Ind}} p\left(P_j^t \mid \boldsymbol{Pa}(P_i)\right).$$

The parameters of each conditional probability distribution resp. conditional probability density (CPD) in the model were estimated from the experimental data provided by CRF. More specifically, let X be a discrete variable, Y be a continuous variable, and $\boldsymbol{U}$ be a set of discrete variables. Each CPD $P(X \mid \boldsymbol{u})$ is modelled as a multivariate distribution with the parameters given by the posterior mean derived from the experimental data using a Dirichlet prior (see Koller and Friedman, 2009) and each CPD $P(X|\boldsymbol{u})$ is modelled as a Normal distribution with the parameters given by the maximum a-posteriori derived from the experimental data using a Normal-Inverse-Wishart prior (see Murphy, 2012). Figure 35 shows the learned graph-structure factorizing $p(I^t, B^t, \boldsymbol{P}^t | I^{t-1}, B^{t-1}, L^t = \text{slow\_lane})$, Figure 36 shows the learned graph-structure factorizing $p(I^t, B^t, \boldsymbol{P}^t | I^{t-1}, B^{t-1}, L^t = \text{fast\_lane})$.

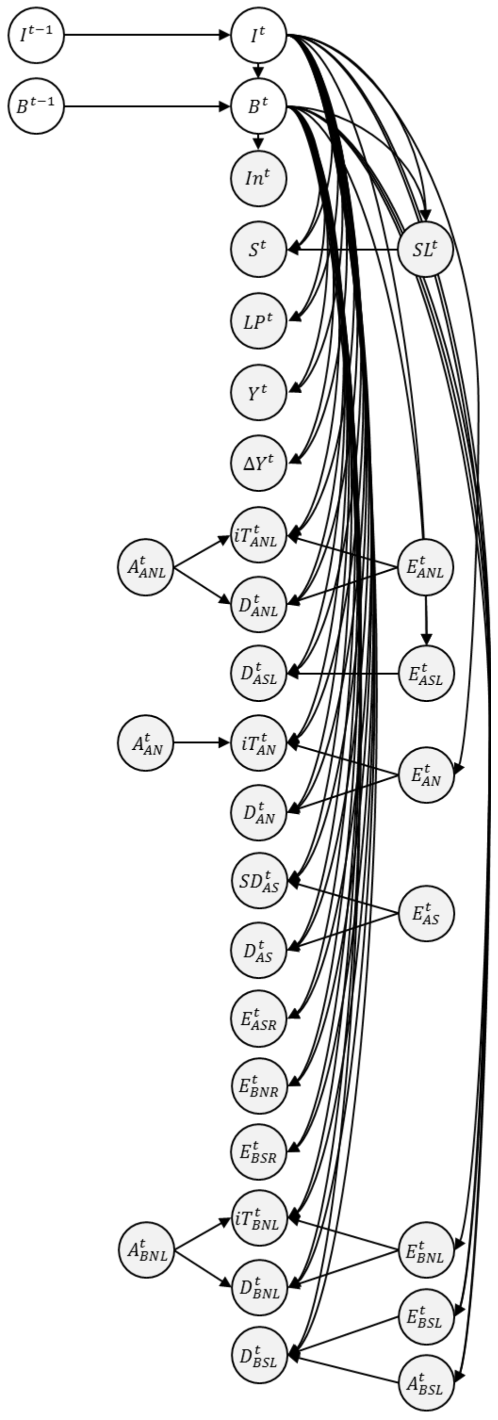
Within the DIR module, the model is used for online intention and manoeuvre recognition, which is achieved by the online-estimation or filtering of the belief state $P(I^t, B^t | l^{1:t}, \boldsymbol{p}_*^{1:t})$, i.e., the probability over intentions and behaviours given all contextual and perceptual evidence collected up to the present time t. Abstracting from the finer factorization of the CPDs in the model, inference can be performed by the usual recursive filtering algorithm in state-space models (Koller and Friedman, 2009). For the first time slice $t = 1$, $P(I^1, B^1 | l^1, \boldsymbol{p}_*^1)$ can be inferred by

$$P(I^1, B^1 | l^1, \boldsymbol{p}_*^1) = \frac{1}{Z} p(\boldsymbol{p}_*^1 | I^1, B^1, l^1) P(I^1, B^1 | l^t),$$

where Z is a normalization constant, so that the probabilities sum to one. For each subsequent time $t > 1$, we can use the former belief state $P(I^{t-1}, B^{t-1} | l^{1:t-1}, \boldsymbol{p}_*^{1:t-1})$ to infer $P(I^t, B^t | l^{1:t}, \boldsymbol{p}_*^{1:t})$ as

$$P(I^t, B^t | l^{1:t}, \boldsymbol{p}_*^{1:t}) \\ = \frac{1}{Z} p(\boldsymbol{p}_*^t | I^t, B^t, l^t) \sum_{i^{t-1} \in I^{t-1}} \sum_{b^{t-1} \in B^{t-1}} P(I^t, B^t | i^{t-1}, b^{t-1}, l^t)\, P(I^{t-1}, B^{t-1} | l^{1:t-1}, \boldsymbol{p}_*^{1:t-1}),$$

where Z is once again a (different) normalization constant. Note that internally, the DIR module uses the slightly more general technique of clique trees for performing these inferences, which however are mathematically equivalent.



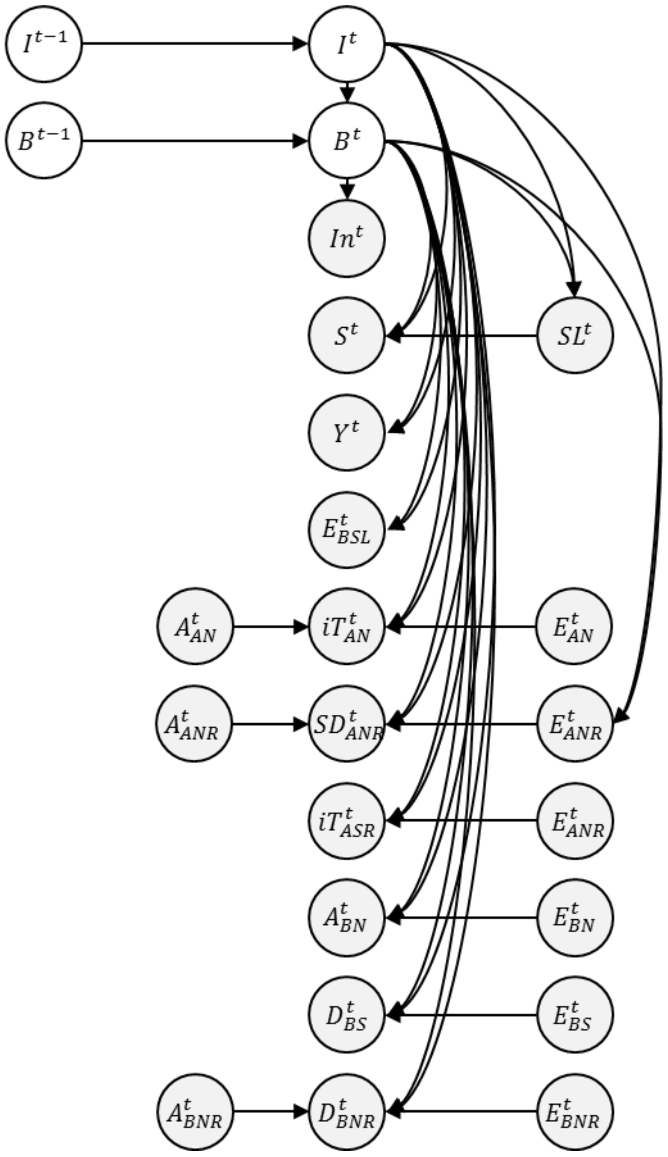
Variables:

I: Intentions of the driver
B: Behaviours/manoeuvres of the driver
In: Indicator signal
S: Velocity of the ego-vehicle
SL: Speed limit
LP: Lateral position of the ego-vehicle in respect to the lane edge of the fast lane
Y: Yaw (or heading) angle of the ego-vehicle
$\triangle Y$: Yaw-rate of the ego-vehicle
E_X: Existence of a vehicle X in the vicinity of the ego-vehicle
A_X: The area (near or far) a vehicle X is inhabiting in respect to the ego-vehicle
iT_X: Inverse time to collision to a vehicle X
SD_X: Speed difference between the ego-vehicle and a vehicle X
D_X: Distance to a vehicle X

Vehicle identifiers:

AN: Lead-vehicle on the slow lane
AS: Lead-vehicle of the lead-vehicle on the slow lane
ANL: Lead-vehicle on the fast lane
ASL: Lead-vehicle of the lead-vehicle on the fast lane
BNL: Following-vehicle on the fast lane
BSL: Following-vehicle of the following-vehicle on the fast lane
ASR: Lead-vehicle of the lead-vehicle on the lane right to the slow lane (entries, exits, and sensor failures)
BNR: Following-vehicle on the lane right to the slow lane (entries, exits, and sensor failures)
BNR: Following-vehicle on the lane right to the slow lane (entries, exits, and sensor failures)
BS: Following-vehicle of the following-vehicle on the fast lane (implying the existence of a following vehicle)

Figure 35: Learned graph-structure representing $p(I^t, B^t, P^t | I^{t-1}, B^{t-1} L^t = \text{slow\_lane})$. The additional parent $L^t = \text{slow\_lane}$ and variables not conditioned by I^t or B^t are omitted to improve visibility.



Variables:
I: Intentions of the driver
B: Behaviours/manoeuvres of the driver
In: Indicator signal
S: Velocity of the ego-vehicle
SL: Speed limit
Y: Yaw (or heading) angle of the ego-vehicle
E_X: Existence of a vehicle X in the vicinity of the ego-vehicle
A_X: The area (near or far) a vehicle X is inhabiting in respect to the ego-vehicle
iT_X: Inverse time to collision to a vehicle X
SD_X: Speed difference between the ego-vehicle and a vehicle X
D_X: Distance to a vehicle X

Vehicle identifiers:
AN: Lead-vehicle on the fast lane
ANR: Lead-vehicle on the slow lane
ASR: Lead-vehicle of the lead-vehicle on the slow lane
BN: Following-vehicle on the fast lane
BS: Following-vehicle of the following-vehicle on the fast lane
BNR: Following-vehicle on the slow lane
BSL: Following-vehicle of the following-vehicle on the lane left to the fast lane (due to sensor errors)

Figure 36: Learned graph-structure factorizing $p(I^t, B^t, P^t | I^{t-1}, B^{t-1} L^t = \text{fast\_lane})$. The additional parent $L^t = \text{fast\_lane}$ and variables not conditioned by I^t or B^t are omitted to improve visibility.

As reported in Deliverable 9.9 "Empirical Evaluation of the Automotive AdCoS and HF-RTP Requirements Definition Update (Feedback)" (D9.9), the learned models have been successfully integrated into the Automotive AdCoS "Adapted Assistance". Based on evaluations of independent test sets, the learned models extend the required accuracy of approx. 80% with an accuracy of 0.908 and provide a mean predictive horizon of 4.19s in the most conservative case, which is significant larger ($p = 2.618^{-11}$) than the current state of the art of approx. 3.0s (for details, please refer to D9.9).

3.7.4 Integration status

The DIR models are used within the DIR *module*, a MTT developed in WP3 and WP9 utilized in the Automotive AdCoS "Adapted Assistance". As the DIR module is a non-lifecycle tool, integration via OLSC has not been realized. Instead, the DIR module has been successfully implemented in terms of RTMaps packages, providing sets of RTMaps components that can be used for AdCoS modelling and utilization in RTMaps. Using RTMaps, the DIR module has been successfully integrated into the AdCoS "Adapted Assistance" and has been tested on the CRF demonstrator vehicle and in the REL driving simulator used for AdCoS evaluation.

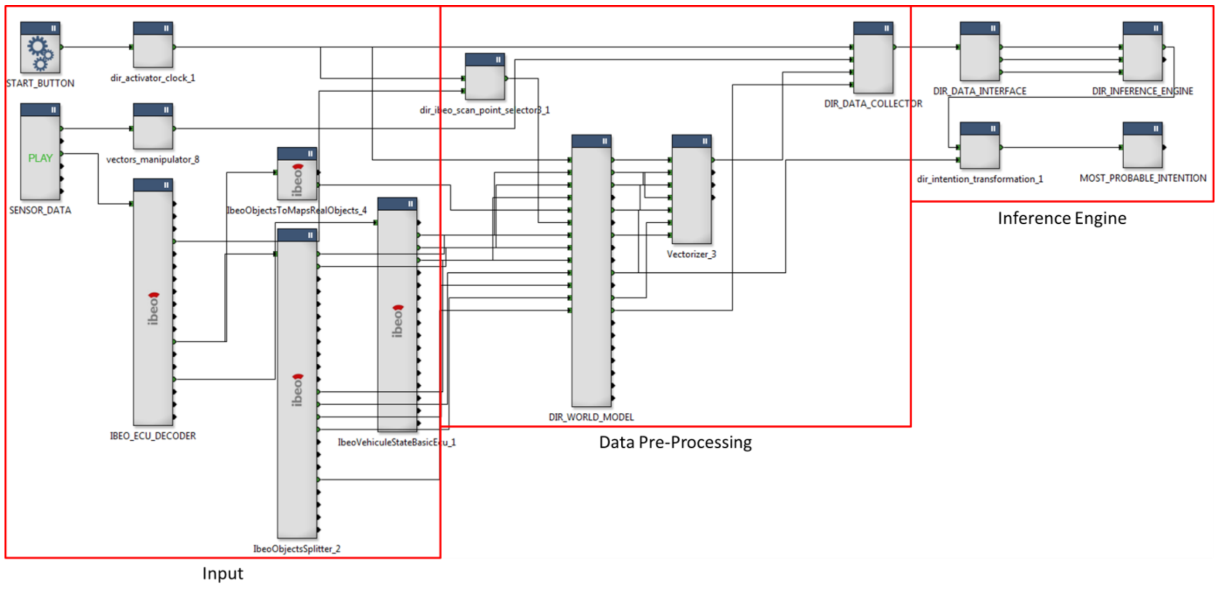
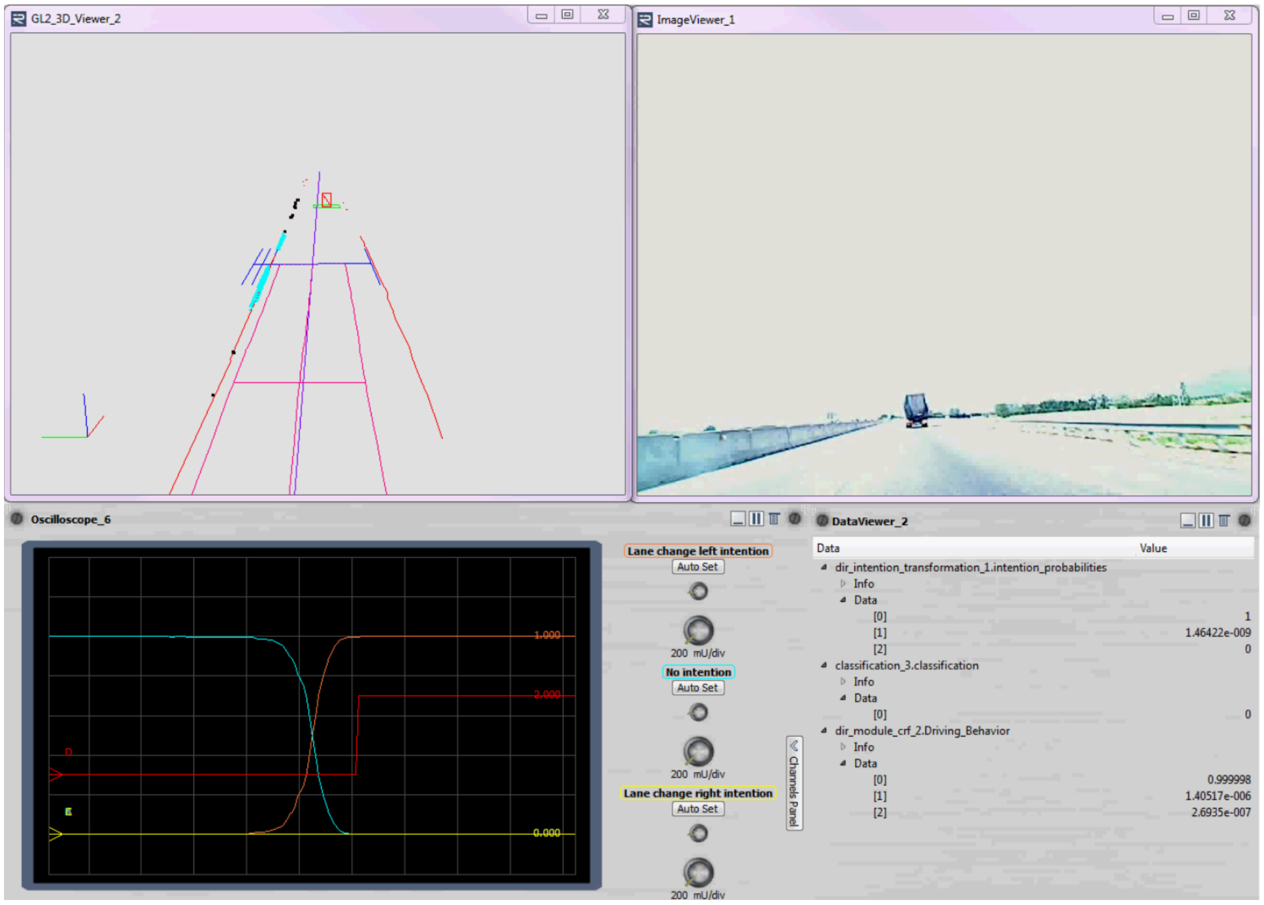


Figure 37: (Simplified) overview of the DIR module integrated in RTMaps, arranged to highlights RTMaps components for the Data Pre-Processing and Inference Engine of the DIR module.

Figure 37 shows an overview of the DIR module modelled in RTMaps, connected to an RTMaps player that provides the experimental data obtained in the CRF demonstrator vehicle. Different DIR models can be selected to be used within the component DIR_INFERENCE_ENGINE. For utilization of the DIR module in the final AdCoS "Adapted Assistance", the player is replaced by RTMaps components providing sensor information in real-time. Note that components dedicated for visualization of the DIR module have been removed to reduce clutter. Figure 38 shows a screenshot of the DIR module during runtime (using pre-recorded experimental data provided by CRF), where the top left image shows a visualization of the data pre-processing component for enhancing the available sensor input and classification of vehicles in the vicinity of the ego-vehicle, the top right image shows the on-board camera installed on the CRF demonstrator vehicle, and the bottom shows a visual and textual summary of the output of the DIR module.



Data	Value
dir_intention_transformation_1.intention_probabilities	
Info	
Data	
[0]	1
[1]	1.46422e-009
[2]	0
classification_3.classification	
Info	
Data	
[0]	0
dir_module_crf_2.Driving_Behavior	
Info	
Data	
[0]	0.999998
[1]	1.40517e-006
[2]	2.6935e-007

Figure 38: Example of driver intention recognition module in RTMaps. In this example, the intention to perform a lane change to the left (bottom, orange line) is recognized approx. 1 sec prior to the activation of the indicator (bottom, red line).

3.8 Cognitive Distraction Classifier (TWT)

3.8.1 State of the art

Distraction is one of the most frequent causes for car accidents (Horberry et al., 2006). During driving, it leads to a delay in recognition of information that is necessary to safely perform the driving task (Regan and Young, 2003). Four different forms of distraction are distinguished while they are not mutually exclusive: visual, auditory, bio-mechanical (physical), and cognitive distraction. While visual and bio-mechanical distraction can be directly observed (e.g. glancing in the rear mirror), cognitive distraction cannot be

directly observed since there is no direct indicator of this process (Liang et al., 2008). While several approaches are suited to measure cognitive distraction, only some of them are applicable for real-time assessments: driving performance (driving dynamics like speed variability as well as driver interactions like breaking), physiological measures like eye, head or facial movements (Metaxas et al., 2004, Recarte et al. 2000; 2003; 2008), and analysis of the voice (Kurniawan et al., 2013).
The identification of cognitive distraction is more complex than that of visual distraction for two reasons. First, because the mechanisms involved in the former have not been described as precisely as that of the latter. Second, because there is no direct indicator of cognitive distraction. The detection of cognitive distraction could presumably be best assessed through the integration of a number of different parameters like eye and face measures of the driver (e.g., blink frequency, pupil size, mouth movements), driving performance measures (e.g., steering wheel movements and breaking behaviour), and auditory signals (e.g., the sound of the driver's voice). Yet, some of the parameters need careful experimental design. For example, mouth opening (yawning) is often taken as a sign of fatigue (Abtahi et al., 2011), but in real driving scenarios, mouth opening can be associated with an ongoing conversation or with singing along to a song in the radio. A combination of several parameters, for example driving performance and mouth opening, might be able to avoid such misattribution (Dong et al., 2011).
Several models for cue integration have been suggested for cognitive modelling of distraction. Machine Learning techniques are a promising approach to investigate several parameters simultaneously (Liang et al. 2007a, b; Zhang et al., 2004). This characteristic is of high importance since it has been shown that changes in the correlation between e.g. eye tracking data and behavioural data can indicate driver distraction (Yekhshatyan et al., 2013). A further crucial advantage of machine learning is that it is adaptive to individual drivers (Zhang et al., 2004). In the context of machine learning, Support Vector Machines (SVMs) and Bayesian Networks have successfully identified the presence of cognitive distraction using eye movements and driving performance (Liang et al. 2007a, b).
The recent dynamic Bayesian model by Liang and Lee (2008) consists of a combined supervised and unsupervised learning approach. In HoliDes, we investigate these approaches, employing different types of data (such as facial video data).

3.8.2 MTT Description

The main idea of the Cognitive Distraction Classifier (CDC) has been described in detail in previous deliverables (e.g., in D2.6, D5.5, D3.7). To avoid redundancy, only a short overview will be provided here.
In the context of HoliDes, TWT developed an applied cognitive and computational cognitive distraction model using different in-car measures (including audio recordings, video face tracking, and behavioural driving parameters) to detect the distraction degree of the driver.
The model takes the driver's auditory and visual perception into consideration and computes his/her distraction degree based on a resource allocation model. This model from Wickens (2002) states that the more a secondary task takes up the same or similar sensory modalities (auditory vs. visual), codes (visual vs. spatial) and processing stages (perceptual, cognitive, responses), the more the secondary task leads to distraction from the primary task. Thus, using the in-car measures, the model will lead to conclusions about the allocation of the driver's resources and therefore enable the computation of his distraction degree. The CDC can be used offline (i.e., post-experimental) or online (i.e., near-to-real-time). During online use, it will provide a continuous interpretation of the cognitive distraction degree computed by a machine learning classifier. The cognitive distraction degree will be identified within a time window of about two seconds. Several machine learning algorithms (Naïve Bayes, Adaptive Boosting, different kinds of Discriminant Analyses) have been evaluated based on an offline analysis of the data. Moreover, different combinations of the feature sets (Audio, Video and Behavioral) have been tested in order to design a proper feature space. The results of the offline analyses mostly serve to test and develop analysis methodology and measurement techniques, and as a validation for the online cognitive model.

The idea behind the CDC was to use it online to classify the driver's cognitive distraction not only during testing of a prototype, but also during everyday interaction with the AdCoS. This online measure of cognitive distraction could in turn be used to adapt the degree of automation of the AdCoS to the driver's cognitive state. Additionally, a safety system could adapt in an appropriate way to a certain level of cognitive distraction. Furthermore, for the integration of the tool into the HF-RTP, its usage during the system validation phase is essential: while interacting with a prototype or some modules of the AdCoS, the operator's degree of cognitive distraction can be determined. Thus, in this context the tool provides feedback whether or not a new system (module) increases or decreases the operator’s degree of cognitive distraction.

3.8.3 Current status

In recent experiments, data have been recorded from participants driving in a driving simulator while being cognitively distracted through a secondary task, namely the n-back task. Video data of the driver’s face, behavioural driving data, and aural (voice) data have been extensively analysed offline. With certain pre-processing techniques, the signal-to-noise ratio was enhanced, and facial features (e.g. eye blink frequency) and driving features (e.g., distance to the pace car, steering jitter) have been extracted. Machine learning algorithms have been deployed and tested for their applicability to classify whether a driver is undistracted (U), slightly distracted (D1) or strongly distracted (D2) given these features. These three level of distraction correspond to the three levels of the chosen n-back task (0-back, 1-back, 2-back) The machine learning classifier first builds a model based on a set of features (e.g. eyebrow position) derived from an experimental training dataset of which the operator state is known since it was modulated using the n-back task. This model-training phase is carried out with a k-fold cross-validation method. Next, the classifier model can be used to classify the current driver’s state as undistracted, slightly distracted or strongly distracted based on the features from a dataset of which the CDC does not know the operators’ state. Recently, the offline classification software has been redeveloped to allow for online analysis.

TWT has been investigating employing audio voice data to determine the level of the driver’s cognitive distraction. Similar to the video data, the audio data are being analyzed based on a selection of relevant features. Currently audio features are being implemented to be used by the machine learning algorithms. In future versions of the CDC, both audio, video, and eye-tracking features will be jointly used to classify the data and to determine the degree of cognitive distraction of a subject. The most recent testing done by TWT, involved a new online version of the CDC.

3.8.4 Integration status

The metrics used to quantify the driver's distraction based on in-car information are developed in T5.2. The cognitive distraction classifier (CDC) has been integrated into the IAS Test Vehicle (WP9). Vehicle data are transmitted from the Ibeo car to the CDC via Ethernet. The CDC output (estimated level of distraction and estimation reliability value) is written to the vehicle CAN, where it is available to the Ibeo autonomous driving system. During periods of increased distraction, autonomous driving can be adapted towards a more defensive style. A first experiment with the IAS Test Vehicle has just been conducted.
Further WP9 AdCoS systems suitable for integration with the CDC are the TAK Simulator AdCoS and potentially the CRF Test Vehicle. TWT has been working towards an integration with TAK, and conducted joint experiments (for details, see D5.6)

Next steps include continuation of collaboration with partners to integrate the CDC in AdCoS, such that the level of automation can adapt to the cognitive state of the driver, or a safety system can interact with the driver in an appropriate way. The integration will also allow us to validate the CDC outside of a simulated environment and access its accuracy during driving in the real world (see D9.10). Further steps are the increase of accuracy for the online version of the CDC and the integration of multiple signals, e.g. auditory signals.

3.9 Pilot Pattern Classifier (TEC)

3.9.1 State of the art

The relationship between situation awareness, mental workload, and performance has been studied in (Nählinder, 2004, 2009; Garland, 200), showing that increasing the mental workload (e.g., with a more demanding task) could lead to a decrease in situation awareness, which could lead in turn to worsen performance.
The analysis of EEG signals has been used in the assessment of the variations of the state of the subjects during the execution of cognitive or sensory-motor tasks, as it is shown in Smith et al. (2001) and Boucsein et al., (2000). It has been also demonstrated (Matousek et al., 1983; Gevins et al., 1990) that EEG is sensitive to variations in vigilance and has been shown

to predict performance degradation because of sustained mental work. On the other hand, it was discovered by Gale et al. (1977) that a decrease in vigilance and deterioration in performance are connected with an increased EEG power spectra in theta band and a change in EEG alpha power. Many studies on EEGs extracted from aircraft pilots were focused on variations in the power of EEG signals in the theta, alpha and beta bands (Dahlstrom et al., 2009; Poythress et al, 2006, Berka et al., 2007 and Dussault et al., 2005).

One of the most famous artificial intelligence techniques to build classifiers working with EEG data is Artificial Neural Networks (ANN). Wilson et al. (2003a) reported achieving, on average, 85.8% classification accuracy for ANNs trained on within-difficulty manipulation types. Wilson et al. (2003b) used ANNs to classify operator state on a multi-task combination. In this case, heart rate (HR), respiration and eye movement measures were used in addition to EEG as ANN inputs resulting in classification accuracies that were 98.5% on average across participants and achieving online classification accuracies ranging from 82%, in the low workload condition, to 86% in the high workload condition.

However, the workload classifiers presented above are subject-specific, that means a new classifier should be trained for each subject and session. In addition, these classifiers do not achieve good results when classifying a group of subjects with the same training dataset and also the same subject in different sessions (Wilson et al., 2010). Under these circumstances, Wang et al. (2012) showed that it is possible to build a classifier based on the hierarchical Bayes model, handling multiple subjects achieving classification accuracies comparable to a specific-subject ANN. Here is also proved that such a performance is stable across three levels of workload, in comparison with ANNs which have been demonstrated to accurately separate no more than two levels of workload.

On the other hand, we can find other researches in which the Support Vector Machines technique (SVM) is used to achieve good results on this field. Borghini et al. (2011) carried out a study on a large sample of aircraft pilots, which was built on theta and alpha frequency increase/decrease ratios with respect to a baseline condition, was demonstrated to correlate with the pilot’s reports on the difficulty of the task performance.

Other techniques such as Random Forest Classifiers (RFC) have been used to identify the sleep stage using EEG data (Fraiwan et al., 2012).

After reviewing the current literature, it can be concluded that the accuracy of the offline detection of the mental states in driver/pilot using EEG, electro-oculogram (EOG) and Heart Rate (HR) is close to 90%.
Some related research works have been carried out lately. This is the case of Dai et al. (2015) in which they worked on cognitive workload discrimination of pilots during flights, or (Bezerianos et al., 2015) where the study aims to look at the difference in coupling of EEG activity of participant pairs while they perform a cooperative, concurrent, independent yet different task at high and low difficulty level. Also Causse et al. (2015) EEG measurements related to instructions showed that increased mental workload was accompanied by lower P3b amplitude. Finally, Johnson et al. (2015) probe-independent algorithms are explored for classifying three levels of task-complexity in a flight simulator experiment.
With respect to the other works presented in the literature, the workload classifier herein presented exploits EEG and eye-tracking data coming from an ad-hoc designed experiment that has been proved to induce cognitive workload effects in a preliminary phase. With the aim of finding a suitable technique that allows a system in the cockpit to be adapted and react as fast as possible depending on the pilots' data, this work proposes the use of RFC and k-Nearest Neighbours (k-NN) as supervised classification techniques, as opposed to ANN and SVM which are techniques usually characterized by higher computational costs. Although this work considers an offline scenario, this is a relevant issue when running on online systems, which motivates undertaking this scenario in future stages of the HoliDes project. In other words, we search for balancing a trade-off between computational cost and acceptable classification ratios. On the other hand, and as opposed to the majority of the researches mentioned in this section, this work recommends and encourages adopting performance scores used in multiclass and imbalance classification problems, such as macro averaged precision and macro averaged recall, beyond the conventional accuracy metric.

3.9.2 MTT Description

Pilot Pattern Classifier tool evaluates a pilot's physiological data and infers performance related properties, e.g. workload and fatigue. As fatigue is a complex term, it is expected that there will be a larger number of sensors for fatigue detection in various bio-signals. The system starts with EEG and eye-tracking sensors, but it is expected to be extended in future.

As a pattern, the indication of raised fatigue is detected and sent as a context annotation to the diversion assistant. As a reaction, the adaptation follows two lines. In the first, the calculation strategy will prefer safety over economy in evaluating the diversion options. It means that airports with easier approach and landing, with better supporting infrastructure or airports the crew is more familiar with will be selected rather than those that are more suitable with respect to fuel savings or service prices.

3.9.3 Current status

Due to the complexity of fatigue detection, the pilot state classifier has been developed as a proof of concept processing/evaluating its data offline. In future developments it will be faced the online challenge. Is it possible to establish the workload level of a pilot on the basis of EEG and eye tracking data? This is the question that HON, TEC and SNV are investigating.

The Pilot Pattern Classifier (PPC) is a machine learning tool that, when in the classification phase, is able to timely recognize the workload level of the pilot on the basis of EEG and eye-tracking information. In the training phase, PPC needs to be trained in a supervised fashion, i.e., it needs to be fed with a labelled dataset providing examples of EEG and eye-tracking data associated with different (and explicitly indicated) workload conditions.
Along the road towards the solution, three main phases were identified:

1. Data collection: The first challenge addressed, indeed, was to gather the labelled dataset. SNV was in charge of such an activity, to be faced by applying the methodology of experimental analysis of cognitive and communication processes.
2. PPC prototype implementation: Once collected the data, the PPC implementers (TEC, HON) was in charge of designing the best suitable machine learning tool able to leverage the provided dataset to the aim of the online detection of the workload level. A proof-of-concept prototype was the output of that phase.
3. PPC performance evaluation: The performance of the developed prototype was assessed in order to establish if the PPC challenge was reached or not.

At the time of writing, Phase 1, Phase 2 and Phase 3 have been completed.

THE EXPERIMENTS

The first problem was to design an experiment where it was possible to measure the workload effects. The cognitive workload can be defined as a condition that can be revealed by the worsening of performance measures in a primary task due to a secondary interfering task (Goldberg, 1998).
Typically, it has been studied experimentally by means of the dual task paradigm where the performances are measured in terms of response times in the execution of both the primary and the secondary task.
According to that, SNV designed a dual-task paradigm experiment conceived to show the workload effects on directed attention processes, such those that are involved when controlling a cockpit.

SNV performed this experiment twice: the first time without recording EEG and eye-tracking data and the second time recording them. This methodology is due to the fact that we want first to prove that we are able to induce different workload effects by controlling the conditions of the experiments. Only then, we have replicated the experiment by adding the EEG and eye-tracking recording and we exploited the findings of the previous analysis to label the collected data.
Experiments were based on a dual task paradigm in which participants involved in a primary visual search task were asked to perform also a secondary interfering task represented by the syntactical transformation of a sentence. Different types of transformation were tested to see the different workload effects. The results of the first round of the experiments, run with 20 students of the Suor Orsola Benincasa University, showed that three well distinct workload effects (low – LW, medium – MW, and high – HW) appeared according to the type of the requested syntactical transformation (condition) of the secondary task. Further details about the experiment and about the results are provided in D3.6, Section 3.1. Once proved to be able to discriminate between different workload levels, SNV repeated the same experiment by recording also EEG and eye-tracking data.

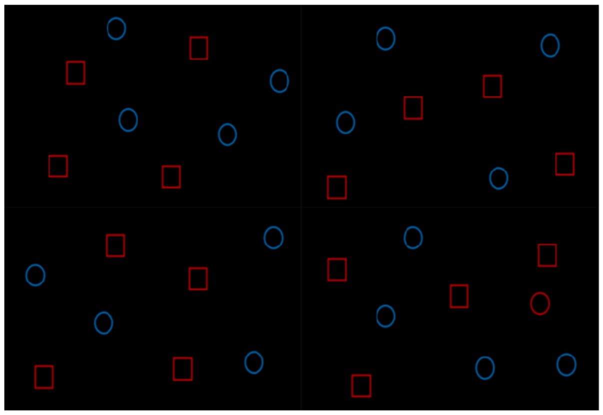
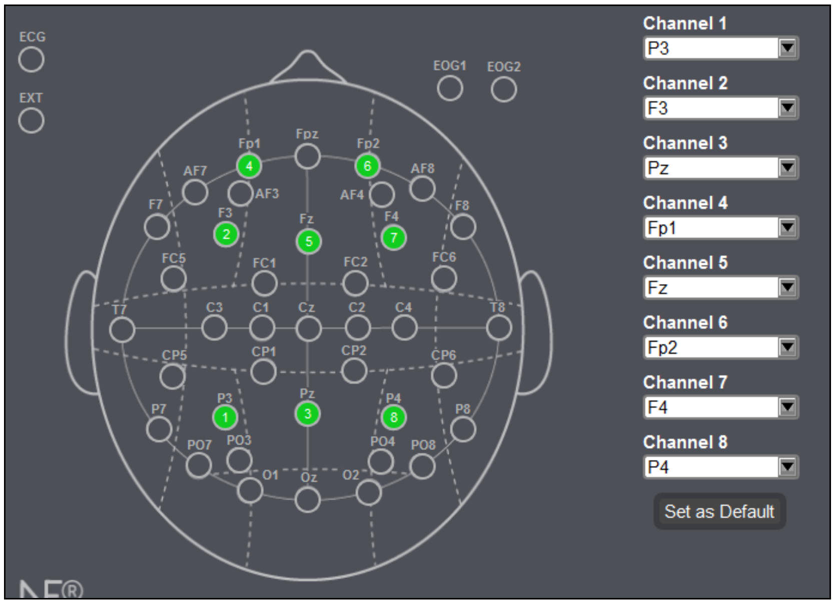


Figure 39: Example of materials used in a visual search task



Channel 1
P3
Channel 2
F3
Channel 3
Pz
Channel 4
Fp1
Channel 5
Fz
Channel 6
Fp2
Channel 7
F4
Channel 8
P4

Figure 40: Position of the electrodes in the PPC experiments

To the aim of the synchronization of the EEG data with the eye-tracking data and the data about the execution of the primary task (digit pressure on the keyboard), the RTMaps platform was provided by INT to SNV. INT provided support to SNV in the development of the RTMaps script needed to perform the synchronized data collection work.

DATA COLLECTION

The RTMaps output dataset needs to be completed with further information coming from DMDX and needed for the labelling: to this aim, a dedicated DMDX parsing program has been developed and applied. In July 2016 a new set of experiments were carried out in Berlin; this will lead us to new experiments and to keep working in our approach, probably facing the online detection challenge and flourishing new collaborations between partners.

To date, the data performed by the experiments of two subjects were exploited to realize the prototype:

- Data set "RecFile_2_20151117_113258_Vectorizer_2_outputFloat" for the user 1 (from now on it will be named "Data set-113258")
- Data set "RecFile_2_20151117_122030_Vectorizer_2_outputFloat" for the user 2 (from now on it will be named "Data set-122030")

These data sets were **supervised**, they showed the specific label for each sample. In our case the possible labels for the class were 3 (**multiclass**): low workload (LW), medium workload (MW), and high workload (HW).

Table 3: Legend of the collected data

Feature	Description
timestamp	time step id, microseconds
alpha_1	alpha power from channel 1 – position P3, micro Volt^2
alpha_2	alpha power from channel 2 – position F3, micro Volt^2
alpha_3	alpha power from channel 3 – position Pz, micro Volt^2
alpha_4	alpha power from channel 4 – position FP1, micro Volt^2
alpha_5	alpha power from channel 5 – position Fz, micro Volt^2
alpha_6	alpha power from channel 6 – position FP2, micro Volt^2
alpha_7	alpha power from channel 7 – position F4, micro Volt^2
alpha_8	alpha power from channel 8 – position P4, micro Volt^2
theta_1	theta power from channel 1 – position P3, micro Volt^2
theta_2	theta power from channel 2 – position F3, micro Volt^2
theta_3	theta power from channel 3 – position Pz, micro Volt^2
theta_4	theta power from channel 4 – position FP1, micro Volt^2
theta_5	theta power from channel 5 – position Fz, micro Volt^2

theta_6	theta power from channel 6 – position FP2, micro Volt^2
theta_7	theta power from channel 7 – position F4, micro Volt^2
theta_8	theta power from channel 8 – position P4, micro Volt^2
r_eye_clos	right eye closure, %
l_eye_clos	left eye closure, %
r_eye_clos_conf	right eye confidence, %
l_eye_clos_conf	left eye confidence, %
eye_clos_calib	eye closure calibration, %
r_pup_dmt	right pupil diameter, mm
l_pup_dmt	left pupil diameter, mm
keyboard	virtual key code of the last pressed digit
item_nr	id number of the current experiment "item", where the item is the couple of the current visual search task and the current transformation task
condition	condition can be: 0 ==> control 1==> from active to passive 2 ==> from passive to active 3 ==> ambiguous phrases
vis_search	0 ==> visual search of the current item has not started yet; 1 ==> visual search of the current item has started
on_keyboard	0 ==> the keyboard has not been pressed yet in the current item 1 ==> the keyboard has been pressed in the current item
label	Workload level of the item condition: LW ==> low MW ==> medium HW ==> high
corr	Correctness 0 => wrong 1 => both tasks performed correctly (spatial transformation and visual search)

MODEL CONSTRUCTION

The model for the Pilot Pattern Classifier was developed in **Python**, using some machine learning libraries such as **sklearn**, and some other scientific libraries such as **numpy**, **pandas**, **scipy**.

ANN (Artificial Neural Networks) and SVM (Support Vectors Machine) are well known techniques applied to EEG data obtaining good results in binary problems. In our case, however, we dealt with a multiclass problem. We used the RFC and k-NN. Both of them are usually applied to multiclass problems. We performed different tests by combining different weights for each class (due to the imbalance problem of the data sets), and by trying with several parameters of the RFC (such as the number of trees, the depth, etc.). We achieved the best results with k-NN when k=3. We used the stratified cross-validation procedure to test the model in order to avoid over-fitting.

The final model was based on the **Nearest Neighbours (k-NN)** technique (with these parameters: k=3), and used 75% of the data sets for training and 25% for testing, using **stratified cross-validation**.

MODEL EVALUATION

When dealing with binary classification, the accuracy of the classification results should be evaluated in terms of standard **classification metrics** such as precision, recall, and F-score. On the other hand, when dealing with a **multiclass classification problem**, like the one we were considering, other metrics should be taken into account, like micro and macro averaged scores. Accuracy is sometimes quite misleading, as you may have a model with relatively 'high' accuracy predicting the 'not so important' class labels fairly accurately but not the classes that are actually critical to the application. In our case, we treat all classes equally, thus our metrics were (i) **macro averaged precision**, (ii) **macro averaged recall**, and (iii) **macro averaged F-score**. See Sokolova et al. (2009) for further information.

RESULTS

In our **first approach** we trained and tested each single model just to know the individual classification performance, and we found F-score ratios close to **0.99**. Being aware of the over-fitting problem in these cases and that this was not conclusive because we were looking for transfer learning, we proceeded with the following approaches.

For the **second approach**, we applied each of the single models to the rest of users trying to find a *universal* user that could transfer their model to the rest. We concluded that user19's model achieved an average F-score of **0.312** for all the users, so we can conclude that this is not enough to consider him as *universal* to transfer his trained model to the rest. In contrast, the less *universal* model belonged to user2 with an average F-score of **0.207**.

At this point it made sense to go on with our **third approach**, in which each user was used once as a test set while the remaining samples formed the training set to build a unique model. In this case we obtained a global average F-score of 0.332. It showed a better score than the second approach but still unsatisfactory from a transfer learning point of view.
The following table shows in detail the classification performance ratios per label, and we can see that MW showed better ratios (recall the number of MW samples were bigger than LW and HW), despite the models were trained in a balance mode.

Table 4: Classification performance per label of the third approach, where each user (1-21) was used once as a test set while the remaining samples formed the training set (P=macro averaged precision, R= macro averaged recall, F= macro averaged F-score)

	LW			MW			HW		
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
1	0,18	0,23	0,20	0,71	0,62	0,66	0,08	0,10	0,09
2	0,12	0,17	0,14	0,66	0,56	0,61	0,04	0,05	0,05
3	0,23	0,08	0,12	0,67	0,84	0,75	0,09	0,05	0,07
4	0,15	0,18	0,16	0,69	0,68	0,68	0,17	0,15	0,16
5	0,11	0,11	0,11	0,72	0,75	0,74	0,23	0,20	0,21
6	0,14	0,12	0,13	0,67	0,73	0,70	0,18	0,14	0,16
7	0,12	0,22	0,16	0,72	0,66	0,69	0,23	0,14	0,17
8	0,11	0,10	0,10	0,71	0,68	0,69	0,19	0,25	0,21
9	0,22	0,21	0,21	0,68	0,69	0,68	0,13	0,12	0,12
10	0,25	0,16	0,19	0,70	0,78	0,73	0,18	0,16	0,17
11	0,17	0,16	0,17	0,69	0,64	0,66	0,17	0,23	0,20
12	0,14	0,20	0,17	0,69	0,68	0,68	0,14	0,09	0,11
13	0,19	0,30	0,23	0,70	0,67	0,68	0,07	0,05	0,06
14	0,22	0,35	0,27	0,72	0,54	0,62	0,15	0,22	0,18
15	0,23	0,25	0,24	0,71	0,68	0,69	0,17	0,19	0,18
16	0,06	0,04	0,05	0,69	0,76	0,72	0,14	0,13	0,14
17	0,05	0,06	0,05	0,69	0,64	0,66	0,19	0,21	0,20

18	0,16	0,14	0,15	0,69	0,75	0,72	0,21	0,14	0,17
19	0,20	0,16	0,18	0,71	0,72	0,72	0,19	0,23	0,21
20	0,12	0,11	0,12	0,68	0,68	0,68	0,20	0,21	0,20
21	0,12	0,11	0,11	0,71	0,70	0,70	0,21	0,23	0,22

Trying to transform the problem into a binary one in our **fourth approach** looking for two different perspectives:

- Firstly, leaving LW (class value 0) as one of the possible values for the target class, and joining MW and HW (class value 1) as the other value for the target class.
- Secondly, leaving HW (class value 2) as one of the possible values for the target class, and joining MW and LW (class value 1) as the other value for the target class.

After that, none of them showed good classification performance ratios, resulting that LW and HW obtained similar ratios to Table 4.

Finally, we decided to divide each data set into three sessions of 4 minutes, in order to try the same approaches than before. Firstly, we trained a model with a session and tested it with another session of the same user, and secondly we made training and testing processes between users' sessions. In both approaches we could not find a global model to be applied to the rest of users with good classification performance ratios.

CONCLUSIONS AND FUTURE WORK

As Borghini et al. (2012) reflect in their review, the difficulty in this field of study lies in the fact that different mental workload levels could correspond to the same driving behaviour, above all when different users are involved in the training process. Some of the existing workload classifiers are subject-specific, meaning a new classifier has to be trained for each subject and session. Some reasons are explained in this paper:

1. It can be seen that the cortical prefrontal areas engagement within the theta frequency band results from the contrast between the two groups of subjects estimated activity indicating that pilots who were naïve to the task relied on the engagement of this area to accomplish it (expert and naïve subjects). Moreover, such differences in the estimated

cortical activity were correlated to the different mental efforts required by the operational task.

2. Rhythms Theta activity increases in the frontal midline with increased task difficulty in younger adults, and to a lesser degree in middle-aged individuals, whereas no increase has been found in older subjects. Alpha activity, in turn, decreases with increasing task difficulty in more widespread areas in older than in younger subjects, whose alpha activity decreases only at the parietal areas. These findings are in good accordance with a previous study that showed an increase in theta activity in the young and a decrease in alpha activity in the elderly during challenging task performance.

In fact, these tools do not achieve good performances when classifying a group of subjects with the same training dataset and also the same subject in different sessions. Here it seems that it would be necessary a tailored model for each pilot. The technique k-NN has revealed itself as really appropriate dealing with EEG data eye-tracking data.

This research has not been able to find a direct transfer learning. Statistical distribution of the data varied across subjects as well as across sessions, even within individual subjects, limiting the transferability of our trained models between them. In addition, the EEG signal is non-stationary, and this is another problem that we should face in our future experiments. As future work, we plan to find some structure in the data that is invariant across datasets (Domain adaptation approach) or some structure in how the same decision rules differ between different subjects or sessions (Rule adaptation approach). In the long term, the HoliDes project will also analyse the possibility of applying our models in an online fashion.

On the other hand, due to the complexity of fatigue detection, the pilot state classifier will be developed as a proof of concept processing/evaluating its data offline. In the long term, the HoliDes project will start working on the field of the online mode, which is a relevant challenge nowadays.

3.9.4 Integration status

Data flow full treatment chain and its interfaces are shown in Figure 41. RTMaps is the central tool to gather data from other tools using its dedicated

interface. Each other data consumer/provider needs to develop a connector to read or send data. The experiment database is connected via a bidirectional interface to allow for data storage and data playback. Tools selected for pilot state assessment work on sensor data streamed from sensors to the tools. These tools either write their results back to RTMaps for data storage in the database or publish the data in a standard form for other consumers. The DivA AdCoS can be one of the consumers when being deployed. During the development, however, the DivA AdCoS connects to RTMaps instead.
Selected tools define two different ways about how they can be connected into the instance of RTP – either as part of RTMaps platform or as OSLC provider/consumer.

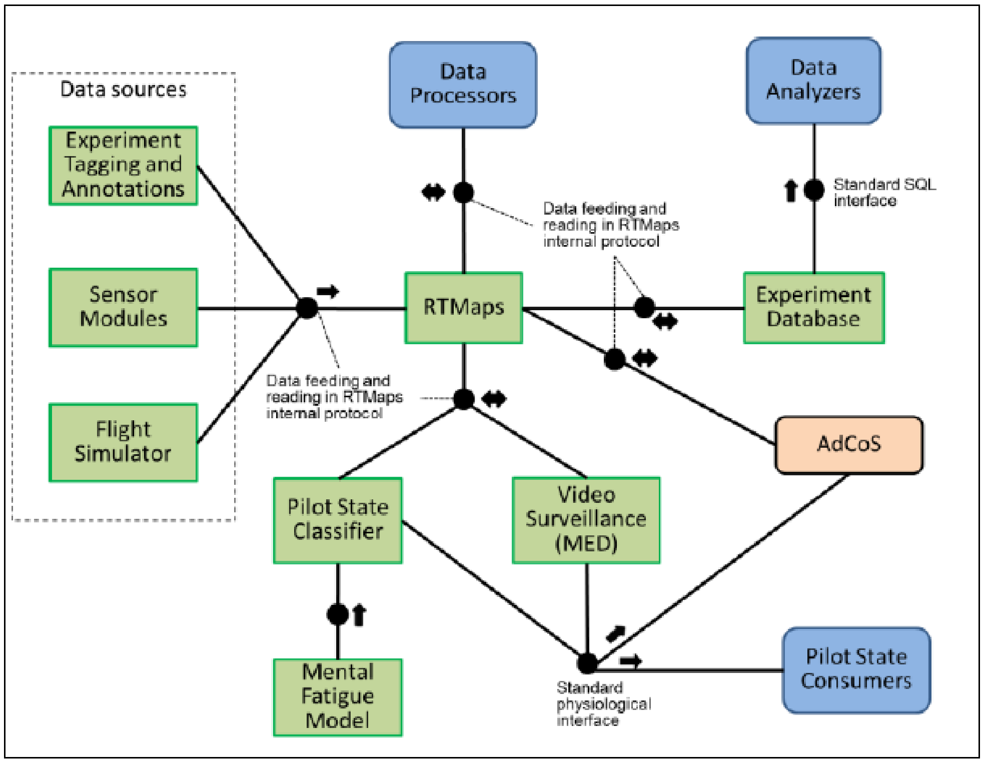


Figure 41: DivA AdCoS data flow and full treatment chain. RTP tools are in green, others in blue. Interfaces (black circles) are described with data transferred through interface and direction (one- or two-way arrows).

For the Pilot Pattern Classifier has been packed as an executable file (in which the AI model is embedded) to make easier the integration with the

rest of tools, and for future developments. This program reads the test data set (from the same folder in which the program is deployed or from a customized path) which is a "csv" file (maintaining the same structure and features than in the training phase, of course).
Then, the program will write the prediction (and its probability) of the test data set in a "csv" file with 2 two fields: 1) prediction (LW-MW-HW), and 2) the probability (%) of each prediction. Finally, this file will can be read by RTMaps and introduced into the full treatment chain, as it is shown in Figure 41.

3.10 Djnn (ENA)

3.10.1 MTT Description

djnn (available at http://djnn.net) is a general framework aimed at describing and executing interactive systems. *djnn* comes with dedicated languages (based on C, C++ or perl) that allows designers of application team to specify a user interface including details independent of modalities (usually called AUI: Abstract User Interface) and details related to modalities (CUI: Concrete User Interface). Designers develop their own djnn components which can be compiled into object files or directly through XML format.

djnn provides also a specific platform dedicated to support execution of applications: the djnn interpreter can be compared to a Java virtual machine, and the djnn XML format to Java byte code. In both cases, executable programs are loaded in memory and run by an interpreter. In this context, the traditional toolkit API of djnn can be considered as an alternative byte code format: components are either stored in XML or in compiled object code.

During the last period, main evolutions of Djnn have been done in the context of WP4 and are related to formal verification and simulation. In order to avoid text duplication, the reader is referred to D4.7 where they are detailed.

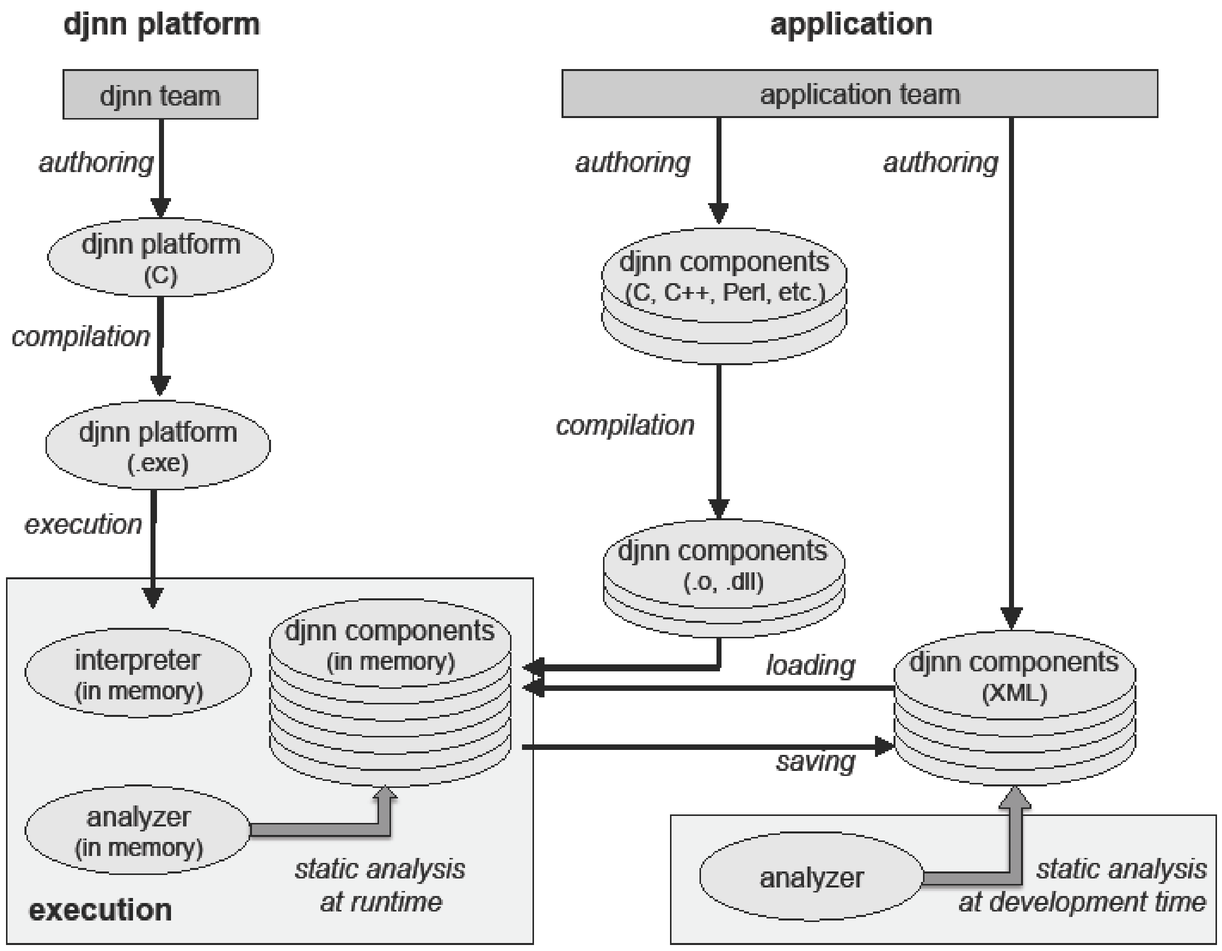


Figure 42: Djnn platform architecture

4 Summary

In the introduction, it was depicted why model-based approaches are essential to facilitate the development of Adaptive and Cooperative Systems. Further, it was described which WPs of the HoliDes project contribute to this goal and how they interact: 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. This models support the modelling of AdCoS used in WP6-9, the implementation of the adaptiveness of the AdCoS themselves in WP3, the guidance of empirical evaluation methods and design of AdCoS in WP5 as well as the formal verification in WP4. Moreover, they also contribute to the common meta-model of the HF-RTP in WP1.
In the subsequent section, an overview of the recent updates of the Task Model, Resource Modelling Language, Human-Machine Cooperation Modelling, Human Operator Models and HMI Models was presented.
In section three, the MTTs of WP2 have been shortly described and their value was explained in the context of the state of the art in this field. Their current development state as well as their integration state have been depicted.

5 References

Abtahi, S., Hariri, B., & Shirmohammadi, S. (2011, May). Driver drowsiness monitoring based on yawning detection. In *Instrumentation and Measurement Technology Conference (I2MTC), 2011 IEEE* (pp. 1-4). IEEE.

Aoude, G. S., Luders, B. D., Lee, K. K. H., Levine, D. S., and How, J. P. (2010). Threat assessment design for driver assistance system at intersections. In: *Proceedings of the IEEE Intelligent Transportation Systems Conference*, pp. 25-30.

Appert, C. and Beaudouin-Lafon, M. (2008). SwingStates: Adding state machines to Java and the Swing toolkit. Journal Software Practice and Experience. 38, 11 (Sep. 2008), 1149-1182.

Berka, C., D. J. Levendowski, M. N. Lumicao, A. Yau, G. Davis, V. T. Zivkovic, R. E. Olmstead, P. D. Tremoulet, and P. L. Craven, "EEG Correlates of Task Engagement and Mental Workload in Vigilance, Learning, and Memory Tasks," *Aviat. Space. Environ. Med.*, vol. 78, no. 5, pp. B231–B244, 2007.

Bezerianos, A., Y. Sun, Y. Chen, K. F. Woong, F. Taya, P. Arico, G. Borghini, F. Babiloni, and N. Thakor, "Cooperation driven coherence: Brains working hard together," in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, 2015, pp. 4696–4699.

Bi, L., Wang, C., Yang, X., Wang, M., and Liu, Y. (2015). Detecting Driver Normal and Emergency Lane-Changing Intentions with Queuing Network-Based Driver Models. In: *Intl. Journal of Human-Computer Interaction*, 31, pp. 139-145.

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

Borghini, G., R. Isabella, G. Vecchiato, J. Toppi, L. Astolfi, C. Caltagirone, and F. Babiloni, "Brainshield: HREEG study of perceived pilot mental workload," *Ital. J. Aerosp. Med.*, vol. 5, pp. 34–47, 2011.

Borghini, G., L. Astolfi, G. Vecchiato, D. Mattia, and F. Babiloni, "Measuring neurophysiological signals in aircraft pilots and car drivers for the assessment of mental workload, fatigue and drowsiness.," *Neurosci. Biobehav. Rev.*, pp. 1–18, 2012.

Boucsein, W. and R. W. Backs, "Engineering psychophysiology as a discipline: Historical and theoretical aspects," *Eng. psychophysiology. Issues Appl.*, pp. 3–30, 2000.

Carsten, O. (2007). From Driver Models to Modelling the Driver: What do we Really Need to Know About the Driver. In: *Modelling Driver Behavior in Automotive Environments*, London: Springer, pp. 105-120.

Causse, M., E. Fabre, L. Giraudet, M. Gonzalez, and V. Peysakhovich, "EEG/ERP as a measure of mental workload in a simple piloting task," *Procedia Manuf.*, vol. 3, pp. 5230–5236, 2015.

Chatty, S. (1994). Extended a graphical toolkit for two handed interaction. *Proceedings of the ACM conference on User Interface Software and technologies (UIST'94)*. ACM Press, 1994.

Chatty S., Magnaudet M., Prun D., Conversy C., Rey S., Poirier M.: Designing, developing and verifying interactive components iteratively with djnn. ERTS 2016 Embedded Real Time Software and Systems, pp. 743—751, 27-29 January 2016

Chatty, S. (2004). Extending a Graphical Toolkit for Two-Handed Interaction. *Proceedings of the 17th ACM symposium on User Interface Software and Technology (UIST 2004),* ACM Press, 2004, pp 267-276.

Dahlstrom, N. and S. Nahlinder, "Mental Workload in Aircraft and Simulator During Basic Civil Aviation Training," *Int. J. Aviat. Psychol.*, vol. 19, no. October 2014, pp. 309–325, 2009.

Dai, Z., J. C. Príncipe, A. Bezerianos, and N. V Thakor, "Cognitive Workload Discrimination in Flight Simulation Task Using a Generalized Measure of Association," in *Neural Information Processing*, 2015, pp. 692–699.

Dong, Y., Hu, Z., Uchimura, K., & Murayama, N. (2011). Driver inattention monitoring system for intelligent vehicles: A review. *IEEE transactions on intelligent transportation systems*, *12*(2), 596-614.

Doshi, A., Morris, B. T., and Trivedi, M. M. (2011). On-Road Prediction of Driver's Intent with Multimodal Sensory Cues. In: Automotive Pervasive Computing, pp. 22-34.

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.

Doshi, A. and Trivedi, M. M. (2011). Tactical Driver Behavior Prediction and Intent Inference: A Review. In: *Proceedings of the 14th International IEEE Conference on Intelligent Transportation Systems*, pp. 1892-1897.

Dragicevic, P. and Fekete, J.-D. (2004). Support for Input Adaptability in the Icon Toolkit. Proceedings of the Sixth International Conference on Multimodal Interfaces (ICMI '04), State College, PA, USA, October 13 - 15, 2004. ACM Press, New York, NY, pp 212-219.

Dussault, C., J. C. Jouanin, M. Philippe, and C. Y. Guezennec, "EEG and ECG changes during simulator operation reflect mental workload and vigilance," *Aviat. Sp. Environ. Med.*, vol. 76, no. 4, pp. 344–351, 2005.

Elliott, C. and Hudak (1997), P. Functional reactive animation. In *International Conference on Functional Programming*, pp. 163-173, June 1997.

Fraiwan, L., K. Lweesy, N. Khasawneh, H. Wenz, and H. Dickhaus, "Automated sleep stage identification system based on time–frequency analysis of a single EEG channel and random forest classifier," *Comput. Methods Programs Biomed.*, vol. 108, no. 1, pp. 10–19, 2012.

Gale, A., R. Davies, and A. Smallbone, "EEG Correlates of signal rate, time in task and individual differences in reaction time during a five-stage sustained attention task," *Ergonomics*, vol. 20, no. 4, pp. 363–367, 1977.

Garcia-Ortiz, M., Fritsch, J., Kummert, F., Gepperth, A. (2011). Behavior prediction at multiple time-scales in inner-city scenarios. In: *Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1066-1071.

Garland, D. J. "Pilot Situation Awareness Training in General Aviation," *Hum. Factors Ergon. Soc. Annu. Meet. Proc.*, vol. 44, no. 11, pp. 357–360, 2000.

Gevins, A. S., S. L. Bressler, B. a. Cutillo, J. Illes, J. C. Miller, J. Stern, and H. R. Jex, "Effects of prolonged mental work on functional brain topography," *Electroencephalogr. Clin. Neurophysiol.*, vol. 76, no. 4, pp. 339–350, 1990.
Halbwachs, N. et al. *The Synchronous Data Flow Programming Language LUSTRE.* In Proc. IEEE 1991 Vol. 79, No. 9.

Gibson, J. J. and Crooks, L. E. (1938). A Theoretical Field-Analysis of Automobile Driving". In: *American Journal of Psychology*, 51, pp. 453-471.

Harel, D. (1987). Statecharts: a visual formalism for complex systems. Science of Computer Programming (8), pp. 231-274.

Hoc, J.-M. (2001). Towards a cognitive approach to human machine cooperation in dynamic situations. International journal of human-computer studies, 54(4):509 - 540.

Hoc, J.-M. Human and automation: a matter of cooperation. In HUMAN 07, pages 277-285. Université de Metz, 2007.

Johnson, M. K., J. A. Blanco, R. J. Gentili, K. J. Jaquess, H. Oh, and B. D. Hatfield, "Probe-Independent EEG Assessment of Mental Workload in Pilots," in *7th International IEEE EMBS Conference on Neural Engineering*, 2015.

Klingelschmitt, S., Platho, M., Gross, H.-M., Willert, V., and Eggert, J. (2014). Combining behavior and situation information for reliably estimating multiple intentions. In: *Proceedings of the 2014 IEEE Intelligent Vehicles Symposium*, pp. 388-393.

Kobiela, F. (2011). *Fahrerintentionserkennung für autonome Notbremssysteme*. Springer, 299 pages.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models – Principles and Techniques*. The MIT Press, Cambridge, Massachusetts, London, England.

Kumagai, T. and Akamatsu, M. (2006). Prediction of Human Driving Behavior Using Dynamic Bayesian Networks. In: *IEICE Transactions on Information and Systems*, Vol. E89-D, No. 2, pp. 857-860.

Kumar, P., Perrollaz, M., Lefèvre, S., and Laugier, C. (2013). Learning-based approach for online lane change intention prediction. In: *Proceedings of the 2013 IEEE Intelligent Vehicles Symposium*, pp. 797-802.

Kurniawan, H., Maslov, A. V., & Pechenizkiy, M. (2013, June). Stress detection from speech and galvanic skin response signals. In *Computer-Based Medical Systems (CBMS), 2013 IEEE 26th International Symposium on* (pp. 209-214). IEEE.

Lefèvre, S., Gao, Y., Vasquez, D., Tseng, H. E., Bajcsy, R., et al. (2014a). Lane Keeping Assistance with Learning-Based Driver Model and Model Predictive Control. In: 12th International Symposium on Advanced Vehicle Control, 8 pages.

Lefèvre, S., Vasquez, D., and Laugier, Ch. (2014b). A Survey on Motion Prediction and Risk Assessment for Intelligent Vehicles. In: *Robomech Journal*, 1, 1, pp. 1-14.

Liang, Y., and Lee, J. D. (2008). Comparing Support Vector Machines (SVMs) and Bayesian Networks (BNs) in detecting driver cognitive distraction using eye movements. *Passive eye monitoring: Algorithms, applications and experiments*, 285-300.

Liang, Y., Lee, J. D., Reyes, M. L. (2007). 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.

Liang, Y., Reyes, M. L., Lee, J. D. (2007). Real-time detection of driver cognitive distraction using Support Vector Machines. IEEE Transactions on Intelligent Transportation Systems 8 (2), 340–350.

Liebner, M., Baumann, M., Klanner, F., and Stiller, Ch. (2012). Driver Intent Inference at Urban Intersections using the Intelligent Driver Model. In: *Proceedings of the 2012 IEEE Intelligent Vehicles Symposium*, pp. 1162-1167.

Mandalia, H. M. and Salvucci, D. D. (2005). Using support vector machines for lane change detection. In: *Proceedings of the Human Factors and Ergonomics Society*, 49(22), pp. 1968-1969

Matousek, M. and I. Petersen, "A method for assessing alertness fluctuations from EEG spectra," *Electroencephalogr. Clin. Neurophysiol.*, vol. 55, pp. 108–113, 1983.

Mehler, B., Reimer, B., Dusek, J.A. (2011). MIT AgeLab Delayed Digit Recall Task (n-back). MIT AgeLab white paper 2011-3B.

Metaxas, D., Venkataraman, S., & Vogler, C. (2004). Image-based stress recognition using a model-based dynamic face tracking system. In Computational Science-ICCS 2004 (pp. 813-821). Springer Berlin Heidelberg.

Michon, J. A. (1985). A critical view of driver behavior models: What do we know, what should we do?. In: Evans, L. and Schwing, C. (Eds.). *Human behavior and traffic safety*, New York: Plenum Press, pp. 485-520.

Morris, B. T., Doshi, A., and Trivedi, M. M. (2011). Lane-Change Intent Prediction for Driver Assistance: On-Road Design and Evaluation. In: *Proceedings of the 2011 IEEE Intelligent Vehicles Symposium* (IV), pp. 895-901.

Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. The MIT Press. Cambridge, Massachusetts, London, England.

Myers, B. A. (1990). A New Model for Handling Input. ACM Transactions on Information Systems, Vol. 8, No. 3, pp. 289-320 Proceedings of the Sixth International Conference on Multimodal Interfaces (ICMI '04), State College, PA, USA, October 13 - 15, 2004. ACM Press, New York, NY, pp 212-219.

Navarro, J., Mars, F., Hoc, J.M., Boisliveau, R., & Vienne, F. (2006). Evaluation of human-machine cooperation applied to lateral control in car driving. IEA 2006 Congress. Amsterdam: Elsevier.

Navarro, J., Mars, F., and Young, M.S. (2010). Lateral Control Assistance in Car Driving: Classification, Review and Future Prospects, IET Intelligent Transport Systems.

Nählinder, S., *Flight simulator training: Assessing the potential*. Department of Management and Engineering, Link{ö}pings universitet, 2009.

Nählinder, S., *ACES-Air combat evaluation system*. Command and Control Systems, Swedish Defence Research Agency, 2004.

Nigay, L. and Coutaz, J. (1993). A design space for multimodal systems: concurrent processing and data fusion. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* (CHI '93). ACM, New York, NY, USA, 172-178.

Ohn-Bar, E., Tawari, A., Martin, S. and Trivedi, M. M. (2014). Predicting Driver Maneuvers by Learning Holistic Features. In: *Proceedings of the IEEE Intelligent Vehicles Symposium 2014*, 6 pages.

Oliver, N. and Pentland, A. P. (2000). Driver Behavior Recognition and Prediction in a SmartCar. In: *Proceedings of the IEEE intelligent vehicles symposium*, pp. 7-12.

Poythress, M., C. Russell, S. Siegel, Patrice D. TremouletPatrick Craven, C. Berka, D. J. Levendowski, D. Chang, A. Baskin, R. Champney, K. Hale, L. Milham, and J. Daniel, "Correlation between Expected Workload and EEG Indices of Cognitive Workload and Task Engagement," *Found. Augment. Cogn.*, no. 1, pp. 32–44, 2006.

Ranney, Th. A. (1994). Models of Driver Behavior: A Review of their Evolution. In: *Accident Analysis and Prevention*, 26, 6, pp. 733-750.

Recarte, M. A., and Nunes, L. M. (2003). Mental workload while driving: effects on visual search, discrimination, and decision making. *Journal of experimental psychology: Applied*, *9*(2), 119.

Recarte, M. Á., Pérez, E., Conchillo, Á., & Nunes, L. M. (2008). Mental workload and visual impairment: Differences between pupil, blink, and subjective rating. *The Spanish journal of psychology*, *11*(02), 374-385.

Recarte, M. A., & Nunes, L. M. (2000). Effects of verbal and spatial-imagery tasks on eye fixations while driving. *Journal of Experimental Psychology: Applied*, *6*(1), 31.

Reimer, B., Mehler, B. (2011). The impact of cognitive workload on physiological arousal in young adult drivers: a field study and simulation validation. *Ergonomics*, 54(10), 932-942.

Smith, M. E., Gevins, A., Brown, H., Karnik, A., and Du, R. "Monitoring task loading with multivariate EEG measures during complex forms of human-computer interaction.," *Hum. Factors*, vol. 43, no. 3, pp. 366–380, 2001.

Sokolova, M. and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, 2009.

Sottet, J-B. et al. (2007). Model-Driven Adaptation for Plastic User Interfaces. In Proc. INTERACT 2007, the eleventh IFIP TC13 International Conference on Human-Computer Interaction, pp. 397-410.

Suzuki, K., Jansson, H. (2003). An analysis of driver's steering behaviour during auditory or haptic warnings for the designing of lane depar-ture warning system. JSAE Review, 24, 65-70.

Tay, Ch. (2009). *Analysis of Dynamic Scenes: Application to Driving Assistance*. PhD thesis. Institut National Polytechnique de Grenoble – INPG.

Torkkola, K., Venkatesan, S., and Liu, H. (2005). Sensor Sequence Modeling for Driving. In: Proceedings of the FLARS Conference, pp. 721-727.

Wang, Z., R. M. Hope, Z. Wang, Q. Ji, and W. D. Gray, "Cross-subject workload classification with a hierarchical Bayes model.," *Neuroimage*, vol. 59, no. 1, pp. 64–9, 2012.

Wilson, G. F. and C. A. Russell, "Operator Functional State Classification Using Multiple Psychophysiological Features in an Air Traffic Control Task," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 45. pp. 381–389, 2003.

Wilson, G. F. and C. A. Russell, "Real-time assessment of mental workload using psychophysiological measures and artificial neural networks.," *Hum. Factors*, vol. 45, no. 4, pp. 635–643, 2003.

Wilson, G. F., C. A. Russell, J. W. Monnin, J. R. Estepp, and J. C. Christensen, "How Does Day-to-Day Variability in Psychophysiological Data Affect Classifier Accuracy?," *Hum. Factors Ergon. Soc. Annu. Meet. Proc.*, no. Augmented Cognition, pp. 264–268, 2010.

Zhang, Y., Owechko, Y., & Zhang, J. (2004, October). Driver cognitive workload estimation: A data-driven perspective. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on* (pp. 642-647).

Yekhshatyan, L. & Lee, J.D. (2013). Changes in the correlation between eye and steering movements indicate driver distraction, *EEE Transactions on Intelligent Transportation Systems, 14(1),* pp. 136-145.