





## HoliDes

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





# D4.6 - Techniques and Tools for Model-based Analysis Vs1.8 incl. Handbooks and Requirements Analysis Update

<b>Project Number:</b>	332933
<b>Classification:</b>	Confidential Version
<b>Work Package(s):</b>	WP 4
<b>Milestone:</b>	M5
<b>Document Version:</b>	V0.7
<b>Issue Date:</b>	31.01.2016
<b>Document Timescale:</b>	Project Start Date: October 1, 2013
Start of the Document:	M21
Final version due:	M28
<b>Deliverable Overview:</b>	This document describes Techniques and Tools for Model-based Analysis in WP4 and their integration into a tailored HF-RTP specifically dedicated to WP4 objectives. Last section is dedicated to MTT requirements updating, according to the progress done during the first phase of the project.
<b>Keywords:</b>	Model-based Analysis, Simulation Techniques and Tools, AdCoS Evaluation, Tailored HF-RTP
<b>Compiled by:</b>	
<b>Authors:</b>	Elvio Gilberto Amparore (UTO), Marco Beccuti (UTO), Susanna Donatelli (UTO), Daniel Prun (ENA), Bohuslav Krena (BUT), Bornard (CVT), Daniel Landa (TEC), Sara Sillaurren (TEC), Zdenek Moravek (HON), T. Bellet (IFS), J.C. Bornard (IFS), B. Richard (IFS), D. Gruyer (IFS)
<b>Reviewers:</b>	Bohuslav Krena (BUT), Ian Giblett (Airbus), and Gert Weller(TAK)

	<b>HoliDes</b> <b>Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems</b>	
--	--	---

<b>Technical Approval:</b>	
<b>Issue Authorisation:</b>	
<p>© All rights reserved by HoliDes consortium</p> <p>This document is supplied by the specific HoliDes work package quoted above on the express condition that it is treated as confidential to those specifically mentioned on the distribution list. No use may be made thereof other than expressly authorised by the HoliDes Project Board.</p>	

<b>RECORD OF REVISION</b>		
Date	Status Description	Authors
23.9.2015	Initial version (Version.01)	Marco Beccuti (UTO)
12.10.2015	Update of content for MDP for Co-pilot	Elvio Gilberto Amparore and Marco Beccuti (UTO)
13.10.2015	Customized version " per partner" prepared and distributed	Elvio Amparore and Susanna Donatelli (UTO)
05.11.2015	Updated content for Djin	Daniel Prun (ENA)
12.11.2015	Update on ANaConDA section	Bohuslav Krena (BUT)
18.11.2015	Section 2.3.1 Experiment Database (EDA) update	Daniel Landa (TEC), Sara Sillaurren (TEC), Zdenek Moravek (HON)
15.12.2015	Merging of the partners' contributions	Elvio Gilberto Amparore (UTO)
11.01.2016	Integration on IFS contribution on COSMODRIVE and the V-HCD platform	T. Bellet (IFS), J.C. Bornard (IFS), B. Richard (IFS), D. Gruyer (IFS)
15.01.2016	Integration of contribution from all partners on section 5 (prototype delivery)	Elvio Gilberto Amparore (UTO) and Susanna Donatelli (UTO)
18.01.2016	Identification and list of commonalities and difference w.r.t. D4.5	Elvio Gilberto Amparore (UTO)
19.01.2016	Finalization of Introduction and conclusion and distribution for internal (Intra WP) review	Susanna Donatelli (UTO)

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

21.01.2016	Intra-WP review back, corrections and distribution to external reviewers	Bohuslav Krena (BUT) and Susanna Donatelli (UTO)
27.01.2016	Feedback from external reviewers	Ian Giblett (AIRBUS) and Gert Weller (TAK)
29.01.2016	Corrections implemented and deliverable ready (confidential and public version) for delivery to technical and quality Assurance manager	Susanna Donatelli (UTO)



## HoliDes

Holistic Human Factors Design of  
Adaptive Cooperative Human-  
Machine Systems



## Table of Contents

<b>1</b>	<b>Introduction: WP4 objectives and MTT .....</b>	<b>10</b>
<b>2</b>	<b>Handbook of Techniques and Tools for Model-based Analysis.....</b>	<b>14</b>
2.1	Human Operator Models .....	14
2.1.1	COSMODRIVE, MOVIDA and the V-HCD platform.....	14
2.1.1.1	Description and Objectives.....	14
2.1.1.2	Application domain and Use Cases .....	15
2.1.1.3	Integration in the HF-RTP and interaction with other MTTs.	16
2.1.2	CASCaS (OFFIS) .....	27
2.1.2.1	Description and Objectives.....	27
2.1.2.2	Application domain and Use Cases .....	28
2.1.2.3	Integration in the HF-RTP and interaction with other MTTs.	29
2.1.3	MDP-based Co-pilot (University of Torino) .....	29
2.1.3.1	Description and Objectives.....	29
2.1.3.2	Application domain and Use Cases .....	31
2.1.3.3	Integration with HF-RTP and interaction with other MTTs...	33
2.2	MTTs specifically dedicated to support AdCoS Modelling and their Virtual Simulation.....	37
2.2.1	RTMaps (INTEMPORA) .....	37
2.2.1.1	Description and Objectives.....	37
2.2.1.2	Application domain and Use Cases .....	38
2.2.1.3	Integration in the HF-RTP and interaction with other MTTs.	38
2.2.2	Pro-SIVIC (CIVITEC) .....	40
2.2.2.1	Description and Objectives.....	40
2.2.2.2	Application domain and Use Cases .....	41
2.2.2.3	Integration in the HF-RTP and interaction with other MTTs.	41
2.3	MTTs for AdCoS design, verification and validation .....	43
2.3.1	ANaConDA, Race Detector & Healer, SearchBestie (Brno University of Technology).....	43
2.3.1.1	Description and Objectives.....	43
2.3.1.2	Application domain and Use Cases .....	47
2.3.1.3	Integration in the HF-RTP and interaction with other MTTs.	47
2.3.2	Djnn (ENAC).....	48
2.3.2.1	Description and Objectives.....	48



## HoliDes

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

**HoliDes**

2.3.2.2	Application domain and Use Cases .....	64
2.3.2.3	Integration in the HF-RTP and interaction with other MTTs.	64
2.3.3	GreatSPN (University of Torino) .....	65
2.3.3.1	Description and Objectives.....	65
2.3.3.2	Application domain and Use Cases .....	66
2.3.3.3	Integration in the HF-RTP and interaction with other MTTs.	68
2.3.4	Experiment Data Archive (EDA) .....	69
2.3.4.1	Description and Objectives.....	69
2.3.4.2	Application domain and Use Cases .....	72
2.3.4.3	Integration in the HF-RTP and interaction with other MTTs.	73
References	.....	75

## List of figures

Figure 1: WP4 MTT for AdCoS Design, Verification and Validation.....	12
Figure 2: Functional architecture of the AdCoS based on MOVIDA .....	15
Figure 3: Simulated ADAS for collision avoidance monitored by MOVIDA....	16
Figure 4: Functional architecture of the COSMODRIVE based V-HCD/HF-RTP .....	17
Figure 5: RTMaps diagram for MOVIDA-AdCoS tests with COSMODRIVE ....	18
Figure 6: Pro-SiVIC simulation with COSMODRIVE at the wheel .....	19
Figure 7: COSMODRIVE Perception-Cognition-Action simulations when driving a virtual car simulated with Pro-SiVIC .....	20
Figure 8: Overview of the V-HCD platform, as an example of a tailored HF-RTP based on RTMaps for automotive application .....	21
Figure 9: V-Design process of AdCoS with the COSMODRIVE platform .....	22
Figure 10: Example of critical scenario simulation, due to visual distraction of the driver.....	23
Figure 11: Virtual design and evaluation of MOVIDA-AdCoS with the V-HCD platform .....	25
Figure 12: Example of virtual evaluation of car sensors and embedded algorithms of driving assistance, based on RTMaps and Pro-SiVIC software	26
Figure 13: Structure of the cognitive architecture CASCaS with all the internal components and the major data flows.....	28
Figure 14: internal status of the co-pilot, with the output strategy. ....	32
Figure 15: GreatSPN GUI while playing the token game.....	35
Figure 16: The RTMaps component of the Co-pilot. ....	36
Figure 17: The RTMaps Studio .....	38
Figure 18: An AdCoS development toolchain .....	39
Figure 19: The Pro-SiVIC <sup>®</sup> simulator .....	40
Figure 20: Pro-SiVIC <sup>®</sup> interoperability with RTMaps .....	42
Figure 21: A simple analyser monitoring lock operations.....	44
Figure 22: An example of different noise settings for reads and writes.....	45
Figure 23: A scheme of usage Race Detector & Healer when analysing concurrent Java programs.....	46
Figure 24: djnn platform architecture .....	49
Figure 25: djnn binding component expressed in Petri net .....	55
Figure 26: djnn assignment component expressed in Petri net.....	55
Figure 27: Approach for verification .....	57
Figure 28: djnn RTMaps integration .....	58

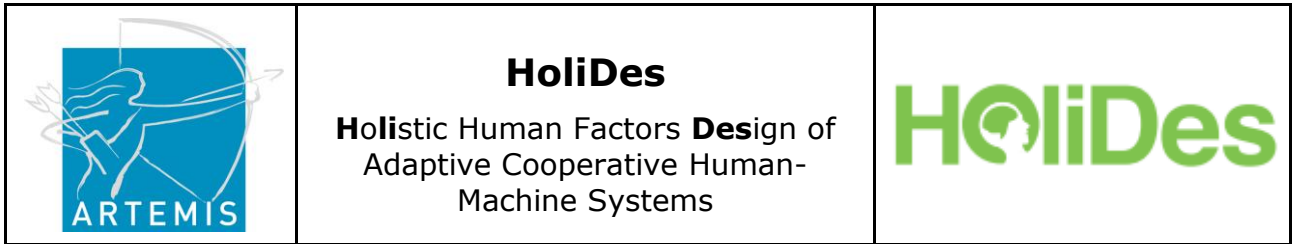


Figure 29 Example of a DDSPublisher component. .... 60

Figure 30 Example of connection with RTMaps..... 62

Figure 31 Structure of the application..... 63

Figure 32: The use of the GreatSPN model-checking facilities ..... 68

Figure 33: EDA data model ..... 71



## HoliDes



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

**HoliDes**

### **Executive Summary**

This document describes Methods, Techniques and Tools (MTTs) for Model-based Analysis in WP4 and their integration into a tailored HF-RTP specifically dedicated to WP4 objectives. Section 1 (introduction) provides a description and a brief recall of WP4 objectives dedicated to AdCoS verification and validation (respectively related with their efficiency and effectiveness). Section 2 provides a description of the main MTTs currently identified to be used in WP4. Section 3 provides a description of the integration plan and the status of the tailored HF-RTP(s) interconnecting WP4 MTTs, in order to support AdCoS validation and verification. Section 4 is dedicated to MTTs' requirements updating, while Section 5 describes prototype delivery, for the MTTs that are already in a distributable form. Finally, the conclusions and perspective section will introduce the work of WP4 for the last period of the project.

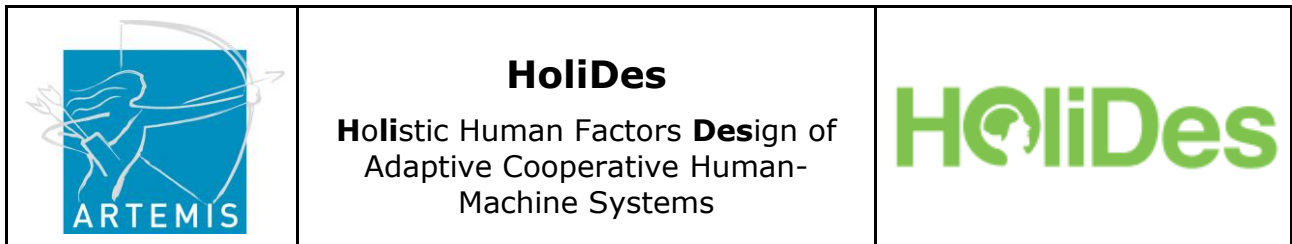


	<p><b>HoliDes</b></p> <p><b>H</b>olistic Human Factors <b>D</b>esign of Adaptive Cooperative Human- Machine Systems</p>	
--	---	---

## **New material with respect to Handbook v1.5 (deliverable 4.5)**

This handbook is an incremental update of the Handbook version 1.5. This is a general summary of the changes made to this document, highlighted in blue:

- The introduction (section 1) which describes the MTT concept and the WP4 objectives has received almost no changes.
- The COSMODRIVE tool section (2.1.1) has been completely rewritten, and now describes COSMODRIVE, MOVIDA and the V-HCD platform.
- The CASCaS framework (2.1.2) description is unchanged and its status is considered stable/complete;
- The MDP co-pilot (2.1.3) has been completely rewritten and describes the current status of the MDP solution and verification algorithms used for the WP9 co-pilot prototype.
- The RTMaps description (2.2.1) has not received updates as its status is stable and RTMaps is already in use.
- The Pro-SIVIC tool description (2.2.2) is unchanged.
- The tool ANaConDA (2.3.1) has received a general overhaul and it is now more detailed.
- The tool djnn (2.3.2) includes a (significantly) more detailed description.
- Description of the GreatSPN tool (2.3.3) is largely unchanged.
- Experiment Data Archive (2.3.4) is a recently developed tool, added to the handbook.
- Section 3 (integration plan) is unchanged, and includes a general description of the HF-RTP platform and OSLC (a more detailed description can be found in WP1 deliverables).
- MTT requirements (section 4) have local updates to reflect the current status of the project.
- A new section (5) has been added to describe the delivery of the MTT prototypes, their availability and the platforms they work on.



## 1 Introduction: WP4 objectives and MTT



According to the objectives stated in the HoliDes description of work, the global aim of WP4 is to “*develop techniques and tools for model-based formal simulation and formal verification of Adaptive Cooperative Human-Machine Systems (AdCoS) against human factor and safety regulations*”.

Verification and validation are two system engineering technical processes (ISO IEC 2008). **Verification** tries to check whether the technical requirements are fulfilled by the system and is related with system **Efficiency**<sup>1</sup> (answering the question “Are we building the system right?”). By contrast, **Validation** deals with users’ task and operational related requirements, trying to check whether the system we are building fulfils according to end user needs (answering the question “Are we building the right system?”) that is more directly related with system **Effectiveness** (i.e. How useful and adequate this system is, regarding end users’ needs, the task they have to performed, and the situational constraints they have to respect).

Although Verification and Validation aim at different objectives (however complementary), they can be supported and implemented by a similar method: **Model-based analysis**. From this approach, the challenge is to construct an intermediate and/or virtual representation of a future system – as a “model” (of the AdCoS, in HoliDes) - and to search for evidences directly on this representation. Models liable to be used in WP4 may describe AdCoS systems at various levels, from high levels of abstraction (functional overview of a global system as a whole, to be used in a large set of use cases, for instance) or, on the contrary, according to low levels of abstraction, including details related to implementation (in particular use case, for instance) and/or by considering individually some specific

---

<sup>1</sup> “While efficiency refers to how well something is done, effectiveness refers to how useful something is. For example, a car is a very effective form of transportation, able to move people across long distances, to specific places, but a car may not transport people efficiently because of how it uses fuel.” ([http://www.diffen.com/difference/Effectiveness\\_vs\\_Efficiency](http://www.diffen.com/difference/Effectiveness_vs_Efficiency)).

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

components of the AdCoS architecture (like sensors, algorithms supporting adaptation and cooperation functionalities, or HMI, for instance).

In addition, the AdCoS model-based Verification and Validation process can be based on two different approaches:

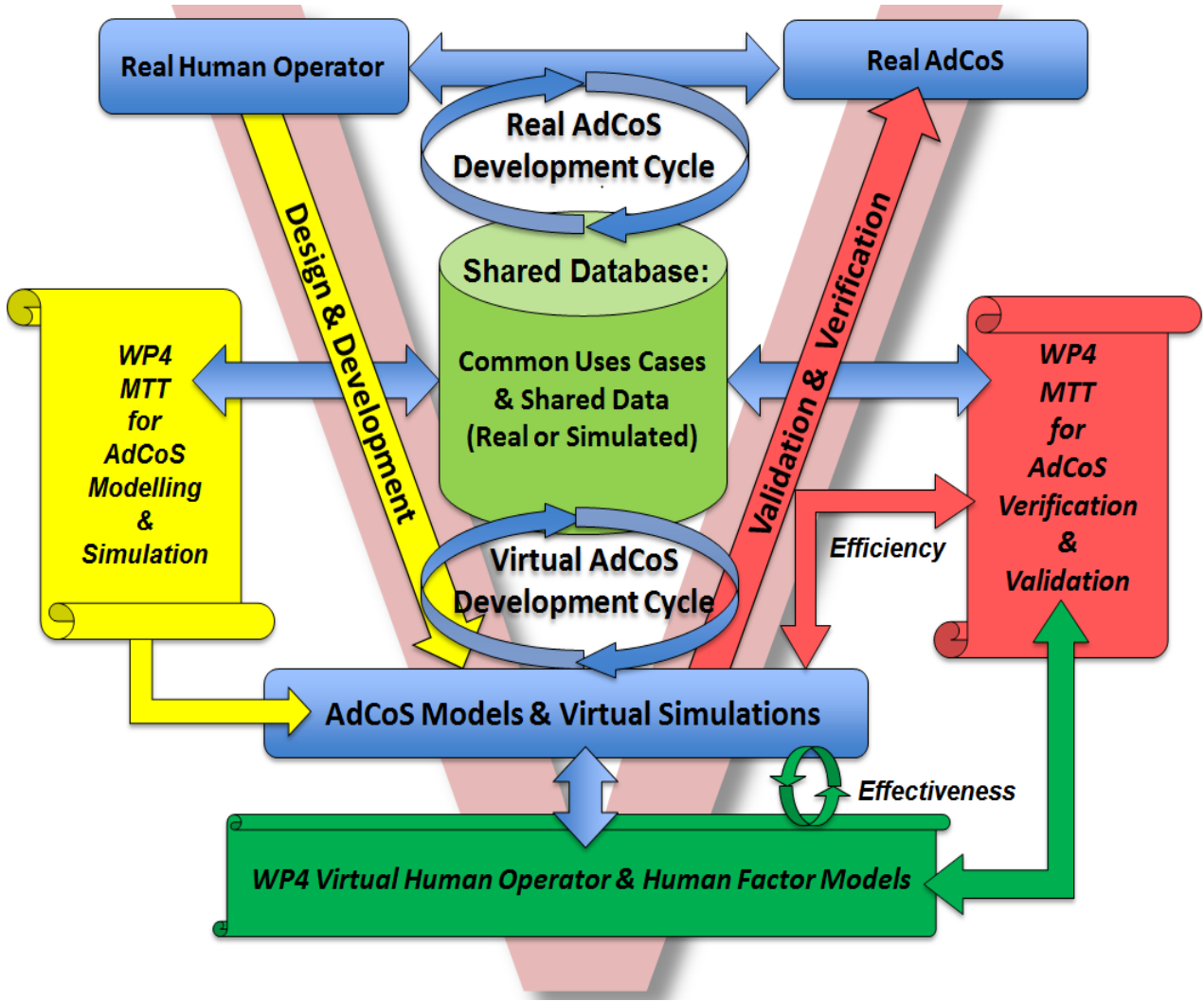
- **Static analysis** of the AdCoS model through mathematical demonstration or formal analysis, like logic diagrams to check the functional architecture of a system, for instance.
- **Dynamic simulation** of the AdCoS model during which dedicated simulations and observations are performed and stored in a database, to be then processed and analysed for checking the validity, the limits and/or the robustness of algorithms for system adaptation or cooperation, for instance.

Moreover, to support AdCoS design, verification and validation tasks, different types of Methods, Techniques and Tools (MTT), mainly model-based, will be used in WP4. Some of them are more specifically dedicated (1) to **model and/or** to computationally **simulate the AdCoS** itself, (2) others are **Human Operator Models** liable to virtually interact with AdCoS models (in order to investigate AdCoS **effectiveness**, for instance) or to be integrated in the AdCoS itself for supporting its adaptive and cooperative abilities (from monitoring function or as a co-piloting system, for instance), (3) and the last type of MTT can be used to check the **correct behaviour of AdCoS** and their technical **efficiency**.

At last, the main challenge in WP4 is also to have a set of combined set of Model-based Techniques and simulation Tools (pre-existing or specifically designed in WP2 and WP3), interconnected in a tailored HF-RTP (jointly defined with WP1) able to support AdCoS Verification and Validation objectives (as a core step of the general design process cycle of these AdCoS), before their integration in real AdCoS and/or in the various Demonstrators to be developed for the different application domains of HoliDes (in WP6-9).





Figure 1 provides an overview of how these different Model-based Techniques and Tools will be combined in WP4 to support design, development, verification and then validation of AdCoS. The approach includes virtual modelling and simulation approaches - based on Human Factor models – as well as real adaptive and cooperative systems to be used by real humans (i.e. end users).



**Figure 1: WP4 MTT for AdCoS Design, Verification and Validation**

At the end of the second year of the project, the MTTs have been identified, built and made available. Some of them still are under development, while

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



others are more mature. However the overall picture of the MTTs is now complete, and there is still time and effort to complete them and to investigate additional possible uses of the WP4 MTTs presented in this deliverable.

As presented in Figure 1, three main types of MTTs used in WP4 can be distinguished:

- 1) Human Factor and Human Operator Models, in charge to model or to simulate end-users of the future AdCoS,
- 2) MTT specifically dedicated to support AdCoS Modelling and their Virtual Simulation,
- 3) MTT more specifically dedicated to AdCoS Verification and Validation issues.

At last, these different MTTs will be interconnected in an HF-RTP, in order to jointly support the global V-Design cycle of AdCoS presented in Figure 1, in terms of evaluation of their *efficiency* and their *effectiveness*.

These three groups of complementary MTTs to be used in WP4 are successively presented in the next section of this deliverable.

	<p><b>HoliDes</b></p> <p>Holistic Human Factors <b>Design</b> of Adaptive Cooperative Human- Machine Systems</p>	
--	--	---

## 2 Handbook of Techniques and Tools for Model-based Analysis

### 2.1 Human Operator Models

Three main human operator models, developed by 3 HoliDes partners (IFFSTAR, OFFIS, University of Torino), have been identified to support WP4 objectives: COSMODRIVE, MOVIDA, the V-HCD platform, CASCaS and the MDP-based Co-pilot.

#### 2.1.1 COSMODRIVE, MOVIDA and the V-HCD platform

##### 2.1.1.1 Description and Objectives

COSMODRIVE is a Cognitive Simulation Model of the Driver developed by IFS in WP2 in order to provide the HF component of a tailored HF-RTP (named V-HCD for Virtual Human Centred Design), to be designed in WP4. The general objective of this computational model is to virtually simulate human drivers' perceptive and cognitive activities implemented when driving a car, through an iterative "Perception-Cognition-Action" regulation loop. Through this regulation loop, COSMODRIVE is able (i) to simulate human drivers visual scanning of the road environment and then to process and integrate the collected visual pieces of information in the Cognition Module, (ii) to simulate core cognitive functions supporting driver's situational awareness (Bellet T. B.-A., 2009) and *decision-making* processes, and (iii) to implement the driving behaviours as decided and planned at the cognitive level, through a set of actions carried out on vehicle commands (pedals and steering wheels), for dynamically progressing into the road environment. As the "HF" component of the V-HCD platform, the aim of COSMODRIVE use in HoliDes is not only to simulate human perception, cognition and behaviour in an optimal way, but also to simulate the "human error" risk, in terms of misperception of event, erroneous situational awareness of the road environment, or inadequate behavioural performance, due to visual distractions (due to a secondary task performed when driving, for instance).

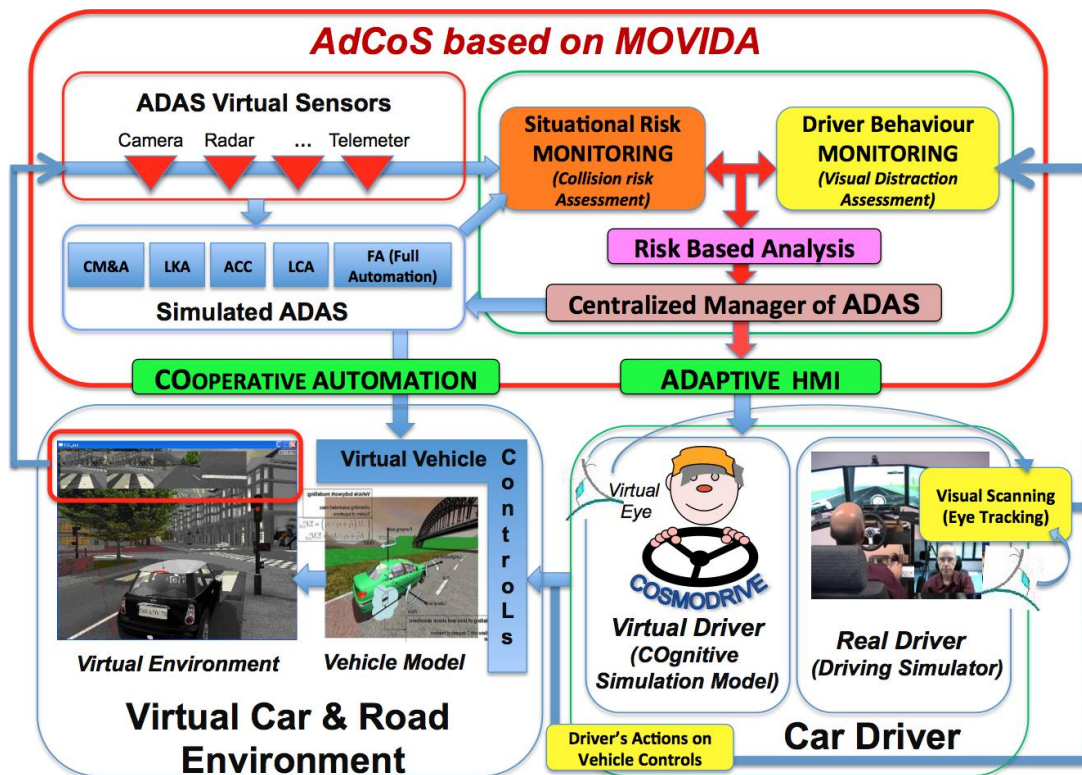
*MOVIDA* (Monitoring of Visual Distraction and risks Assessment) functions are the core of the MOVIDA-AdCoS, an integrated co-piloting system able to supervise and monitor several simulated Advanced Driving Assistance Systems (ADAS) and some state of the driver (e.g. COSMODRIVE perceptive state), which is a component of the Virtual Human Centred Design platform for AdCos development and evaluation in the automotive domain.



**2.1.1.2 Application domain and Use Cases**

COSMODRIVE is dedicated to AdCoS virtual design in the automotive domain. In the frame of WP4 (and then in WP9, as a simulation demonstrator), this model was integrated in the Virtual Human Centred Design (V-HCD) platform of an AdCoS. According to the HoliDes HF-RTP logic, COSMODRIVE plays the role of the Human Factor (HF) component interacting with a virtual AdCoS, also simulated on the HF-RTP. In WP4, the objective is to use this V-HCD/HF-RTP for the virtual design, prototyping and validation of a specific AdCoS developed by IFSTTAR in WP3. This AdCoS is based on *MOVIDA* functions (Monitoring of Visual Distraction and risks Assessment).

Synthetically, *MOVIDA-AdCoS* is an integrative co-piloting system supervising several simulated Advanced Driving Assistance Systems (ADAS) for Collision Avoidance and Lane Change Assistance, to be centrally managed in an adaptive and cooperative way by *MOVIDA* algorithms, according (1) to the drivers' visual distraction states and (2) to the situational risks assessment, as presented in the following Figure.



**Figure 2: Functional architecture of the AdCoS based on MOVIDA**

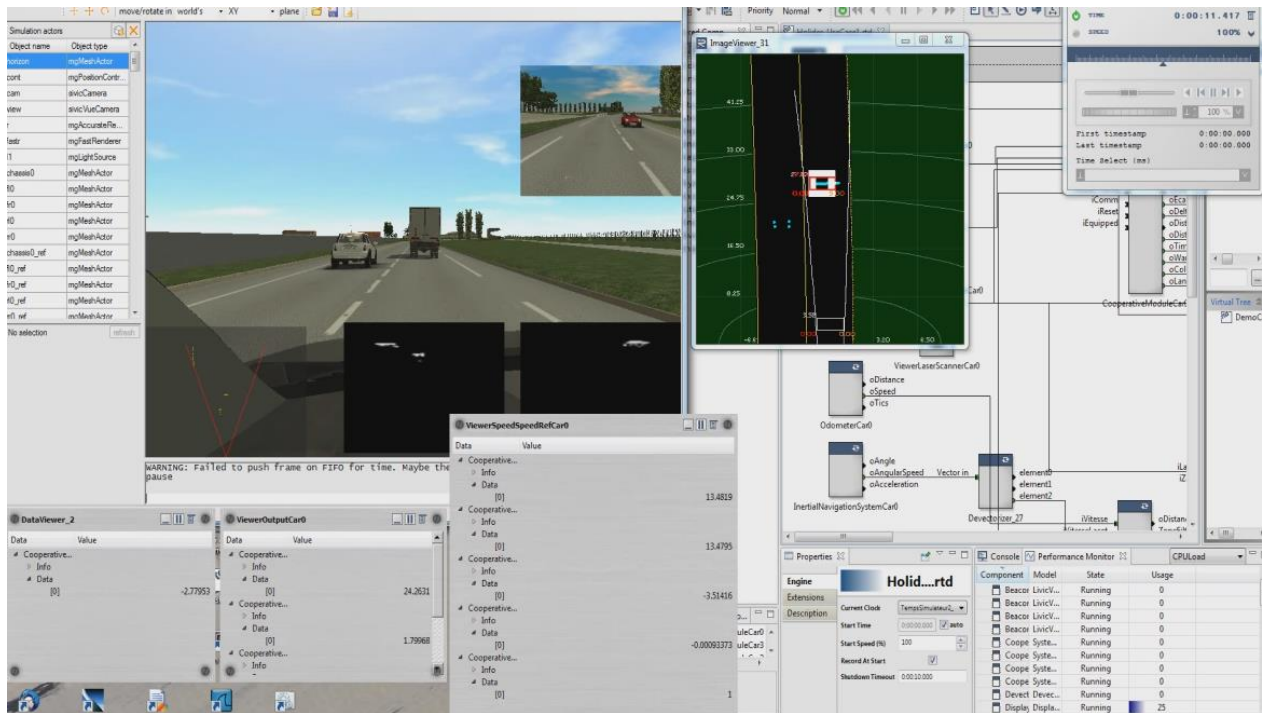


# HoliDes

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



In addition, the following figure presents an example of simulated ADAS (with RT-Maps and Pro-SIVIC), dedicated to Collision Avoidance (Front, Lateral and Rear risks) and Lane Change assistance (LCA), to be monitored by MOVIDA to support an adaptive and cooperative management of human machine interactions.



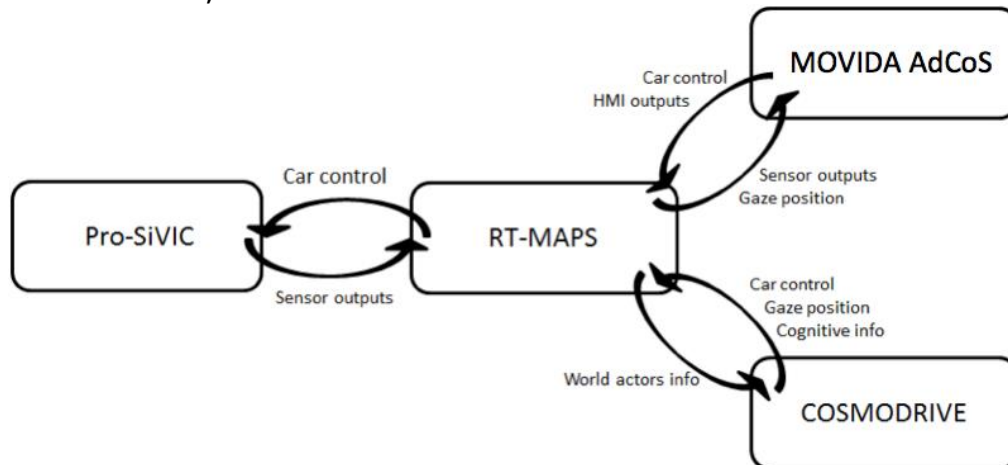
**Figure 3: Simulated ADAS for collision avoidance monitored by MOVIDA**

### 2.1.1.3 Integration in the HF-RTP and interaction with other MTTs

In order to have a Virtual Human Centred Design platform for MOVIDA-AdCoS design and evaluation, COSMODRIVE was connected with RTMaps and Pro-SiVIC. The aim is to have (1) a human driver model (i.e., COSMODRIVE) with a virtual eye (simulating real drivers' visual scanning and visual distraction risk) able to drive (2) a virtual car (simulated with Pro-SIVIC) equipped with (3) several ADAS supervised by the AdCoS manager based on MOVIDA functions (simulated with Pro-SIVIC and RTMaps) into (4) a virtual 3D road environment (simulated with Pro-SIVIC). The following figure provides an overview of the functional architecture of this tailored HF-RTP



tool chain specifically designed for HoliDes to support the MOVIDA-AdCoS virtual design and test. In this platform, RTMaps plays a key role for connecting the different MTTs, supporting ADAS and AdCoS simulation, and allowing COSMODRIVE to perceive the road environment in Pro-SIVIC, to drive the virtual car, and to interact with the AdCoS.



**Figure 4: Functional architecture of the COSMODRIVE based V-HCD/HF-RTP**

To support the interoperability of COSMODRIVE and MOVIDA-based AdCoS with other MTTs developed in HoliDes, these simulation models are interfaced with RTMaps. Indeed, RTMaps offers an easy way to connect multiple tools and data sources. For the HoliDes project, several partners already use it, and RTMaps can be used to interact with COSMODRIVE and MOVIDA.

The RTMaps diagram presented in the figure above provides an overview of the COSMODRIVE and MOVIDA integration/interfacing with this software (and then, with all other MTTs connected with RTMaps). In this figure, the MOVIDA-AdCoS sub-diagram receives on the one hand **inputs** (1) from COSMODRIVE regarding both drivers' visual behavior (to assess 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. On the other hand, MOVIDA-AdCoS generates **outputs** towards Pro-SIVIC virtual car to implement COSMODRIVE and/or MOVIDA-AdCoS actions.



# HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

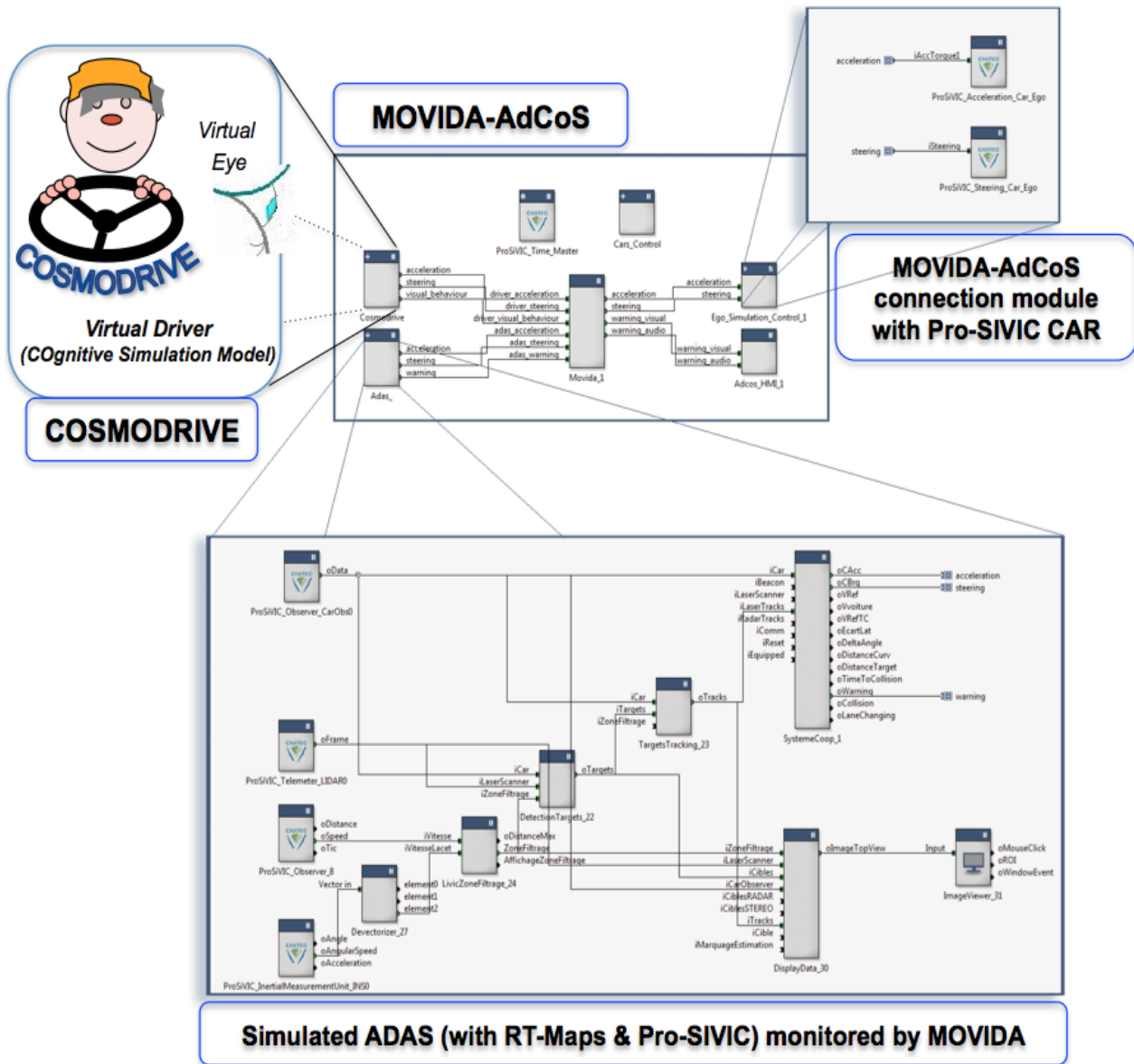


Figure 5: RTMaps diagram for MOVIDA-AdCoS tests with COSMODRIVE

In WP4 and WP9, the objective is to use COSMODRIVE-based simulations on the V-HCD platform 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 6 shows the simulated road environment and the virtual cars: the reference ego car and the lead car in front, both simulated with Pro-SiVIC. This “virtual eye” view is provided by a Pro-SiVIC camera placed at the driver’s head position in the ego car, when COSMODRIVE is following a black car.



**Figure 6: Pro-SiVIC simulation with COSMODRIVE at the wheel**

Figure 7 provides an overview of COSMODRIVE simulation results with the COSMODRIVE/RTMaps/Pro-SiVIC tool chain. The top-right view corresponds to the e road environment simulated with Pro-SiVIC, from the car driver’s point of view. The top left view shows COSMODRIVE’s Perceptive Representation of this road environment, as perceived from the Virtual Eye (the red square section corresponding to the foveal vision of this virtual eye).

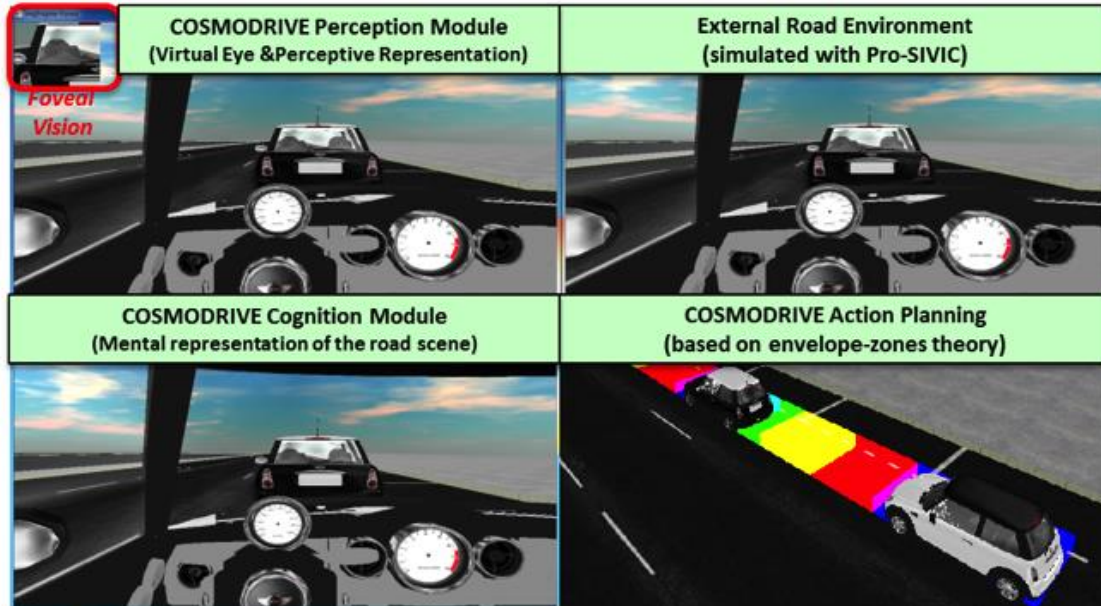
The bottom left view corresponds to COSMODRIVE’s Mental Representation of the road environment, as perceived and understood by the model at this time (i.e. its current Situation Awareness), and the bottom right view illustrates the envelope-zone theory used in COSMODRIVE to support Decision Making and Action Planning, and then driving behaviours implemented by the model in the Pro-SiVIC virtual environment (Bellet T. M., 2012).



## HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

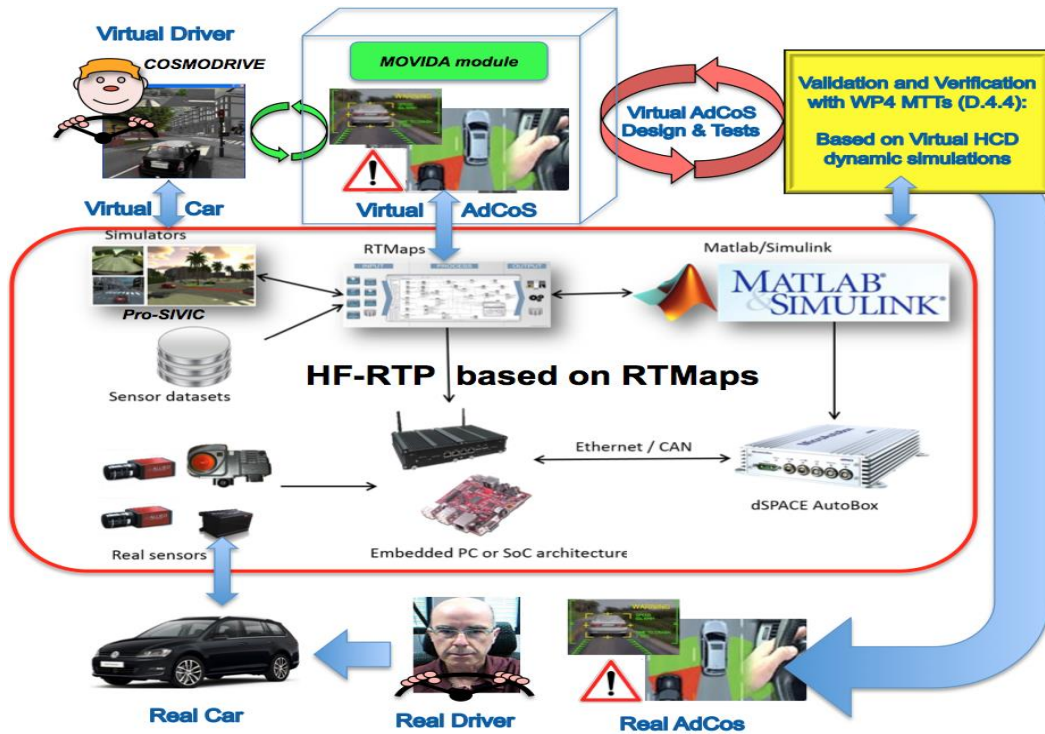
# HoliDes



**Figure 7: COSMODRIVE Perception-Cognition-Action simulations when driving a virtual car simulated with Pro-SiVIC**

This COSMODRIVE/RTMaps/Pro-SiVIC tool chain is fully operational to progressively integrate pieces of information extracted in the road environment at the different levels of the human perceptive and cognitive system. Moreover, the current RTMaps diagrams previously presented in Figure 5 can be easily extended to interact with other HoliDes MTTs, or for interfacing other HoliDes MTTs with Pro-SiVIC and COSMODRIVE.

Regarding the V-HCD platform as a whole, all the MTTs used for supporting the MOVIDA-AdCoS design process are now integrated in RTMaps. Figure 8 provides an overview of the V-HCD based on RTMaps, for supporting the MOVIDA-AdCoS virtual design and evaluation, and then, its potential transfer towards real cars, by means of some RTMaps functionalities.



**Figure 8: Overview of the V-HCD platform, as an example of a tailored HF-RTP based on RTMaps for automotive application**

In this approach, RTMaps, Pro-SiVIC and the V-HCD platform can be used to support virtual simulations of car sensors, ADAS and AdCoS, interacting with a driver model (COSMODRIVE). Simulated data, or data collected on real cars could be used to further design MOVIDA-AdCoS algorithms, according to RTMaps interoperability with other WP9 demonstrators. Connection with other MTTs in WP4, like djnn or GreatSPN, may be also used to support AdCoS Verification and Validation at a virtual level in association with RTMaps dynamic simulation functionalities and/or through the shared database of collected/simulated driving data. At last, RTMaps also provides a support for the transfer of virtual AdCoS toward real cars and/or for empirical data sharing with IAS, CRF and TAK demonstrators.

Figure 9 presents an overview of the V-HCD platform use to support the virtual design, prototyping and evaluation of the MOVIDA-AdCoS. In this “V design process”, only the steps integrated under the red line will be effectively implemented by IFSTAR during the HoliDes project, with the aim of validating and demonstrating in WP9 the advantage of using a tailored HF-



# HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems



RTP to support AdCoS virtual design in automotive domain (i.e., no AdCoS for real cars will be developed by IFSTTAR).

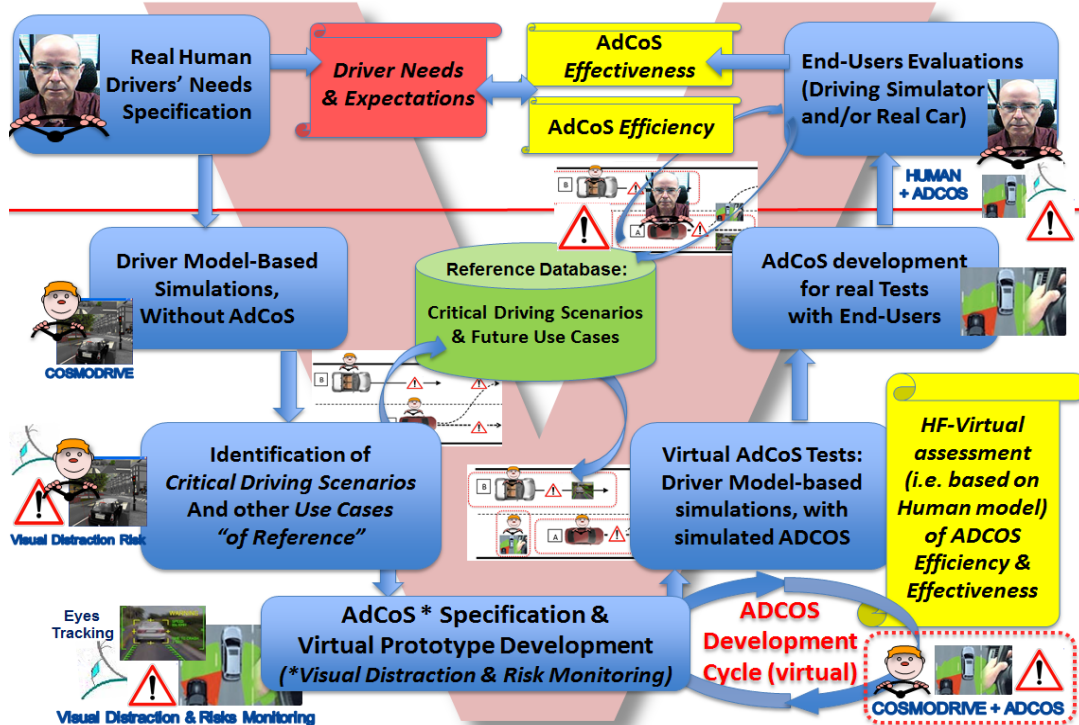


Figure 9: V-Design process of AdCoS with the COSMODRIVE platform

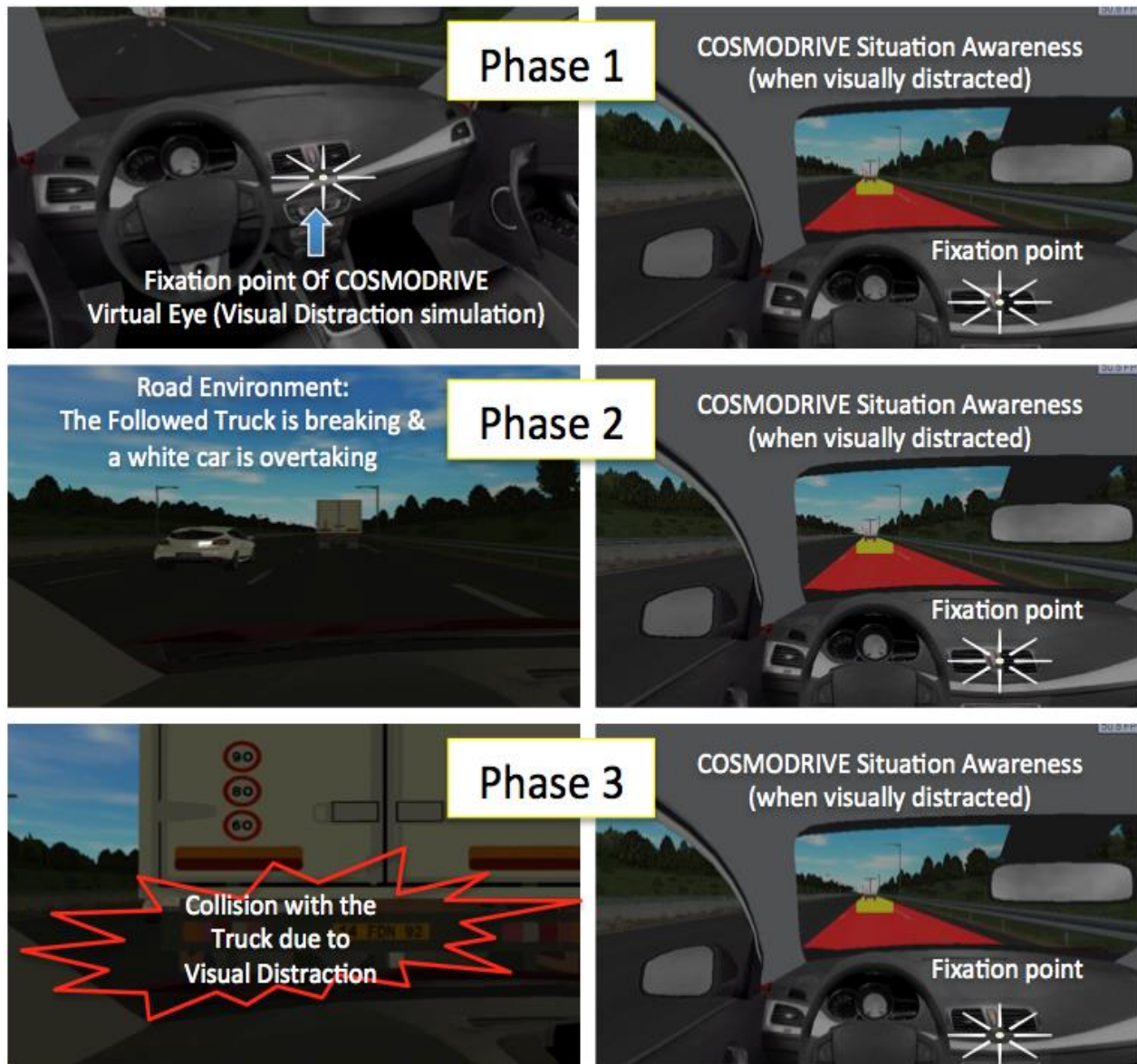
In this Virtual Human Centred Design approach, it is expected to better integrate end-users needs from the earliest stages, and then to support AdCoS efficiency and effectiveness testing in a virtual way, before the development of a real prototype.

The V-HCD platform will be used to simulate driving performances of human drivers *with* and *without* driving assistance (from normal behaviours to critical behaviours due to visual distraction), in order to support the AdCoS and MOVIDA functions virtual design and evaluation at two main levels.



At the earliest stages of the design process, COSMODRIVE-based simulations will be used to estimate human drivers' performances and risks in case of unassisted driving, in order to identify critical driving scenarios liable to be supported by the MOVIDA-AdCoS. These critical scenarios will correspond to

traffic situations for which the visual distraction could critically impact the human drivers' reliability and increase the risk of an accident.

Figure 10 provides a typical example of a "critical scenario" due to visual distraction of the driver, as simulated with COSMODRIVE, when driving without any driving assistance.



**Figure 10: Example of critical scenario simulation, due to visual distraction of the driver.**

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

At phase 1, the visual distraction is starting, and the fixation point of COSMODRIVE virtual eye is inside the car. At this time, the driver's mental model (i.e. situation awareness) of the road environment is still correct because the off-road glance is only starting. However, at phase 2 (left panel), a white car is overtaking the driver and the followed truck (the truck being followed by the ego car) starts to brake. Due to the visual distraction, the situation awareness of COSMODRIVE is not updated (not any overtaking car and no awareness of the braking truck), as depicted in the right panel of phase 2. At phase 3, the crash is close (left panel), and the driver/COSMODRIVE (right panel) is not aware at all of this risk.

Through these types of simulations, it is possible to generate with the V-HCD platform a set of critical scenarios liable to occur in case of visual distraction. Then, all this scenarios may be stored in a reference database, in order to specify a MOVIDA-AdCoS system able to support the driver in this driving context and avoid this type of accidents. Then, during the steps of virtual design process of the MOVIDA-AdCoS, this reference database associated with visual distraction simulations (based on COSMODRIVE) will be used to develop and progressively increase the MOVIDA-AdCoS *efficiency* (i.e., AdCoS developments & virtual tests cycle in Figure 9) in accordance with the different variations of the critical scenarios previously identified.

Figure 11 provides an example of such V-HCD simulation, integrating COSMODRIVE driver, a virtual road environment, and a simulated MOVIDA-AdCoS, for the critical scenario previously presented in Figure 10. Phase 1 is exactly the same as in Figure 10. At phase 2, MOVIDA-AdCoS computes on one side (from the simulated radars) the Time to Collision with the followed truck and also detect an approaching car in the blind spot area (left panel). From the other side, MOVIDA functions also diagnoses the visual distraction status of COSMODRIVE, as represented in the right panel. Consequently, the MOVIDA-AdCoS generates several warning (visual and auditory) informing the driver of the necessity to (1) look at the road, (2) keep the lane, and (3) brake. At phase 3, the warned COSMODRIVE immediately focuses its virtual eye on the road environment and updates its situational awareness (right panel). 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.





## HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

# HoliDes

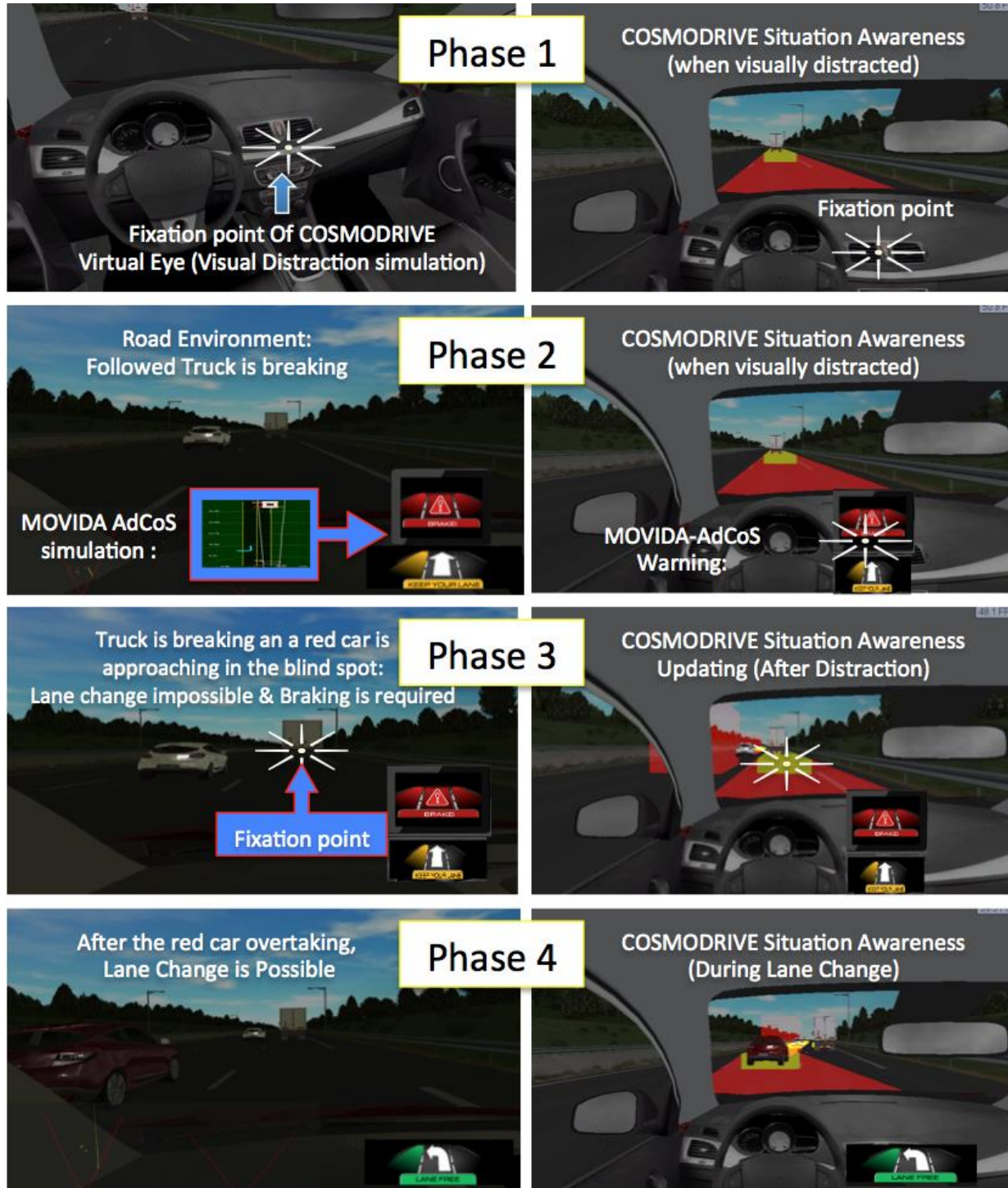


Figure 11: Virtual design and evaluation of MOVIDA-AdCoS with the V-HCD platform



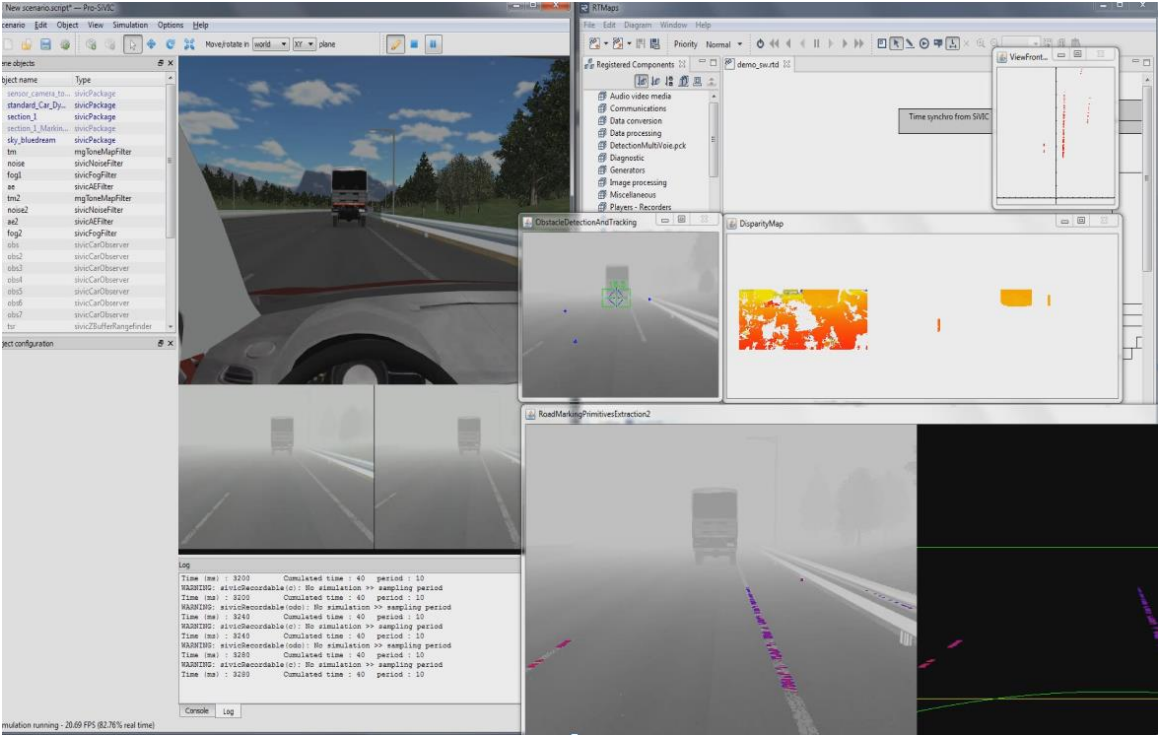
# HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems



Such type of full simulation based on simulated "COSMODRIVE + MOVIDA-AdCoS + ADAS + Car Sensors" will also allow the designer to progressively (i) evaluate the *efficiency* of MOVIDA-AdCoS (i.e. how well does this driving assistance work?) and (ii) to assess its effectiveness of this AdCoS according to human drivers needs and difficulties (i.e. how useful is this driving assistance?), before developing a real prototype for full scale tests with end-users implemented on driving simulators and/or with real cars (final stage of the design process).

Regarding progressive evaluation of the MOVIDA-AdCoS efficiency, it is possible for instance to use Pro-SIVIC and RTMaps (i.e. I-Deep functionalities) to test and validate the robustness of a collision avoidance ADAS monitored by MOVIDA, for instance and its algorithms based on stereo-vision in case of fog (Figure 12).



**Figure 12: Example of virtual evaluation of car sensors and embedded algorithms of driving assistance, based on RTMaps and Pro-SIVIC software**

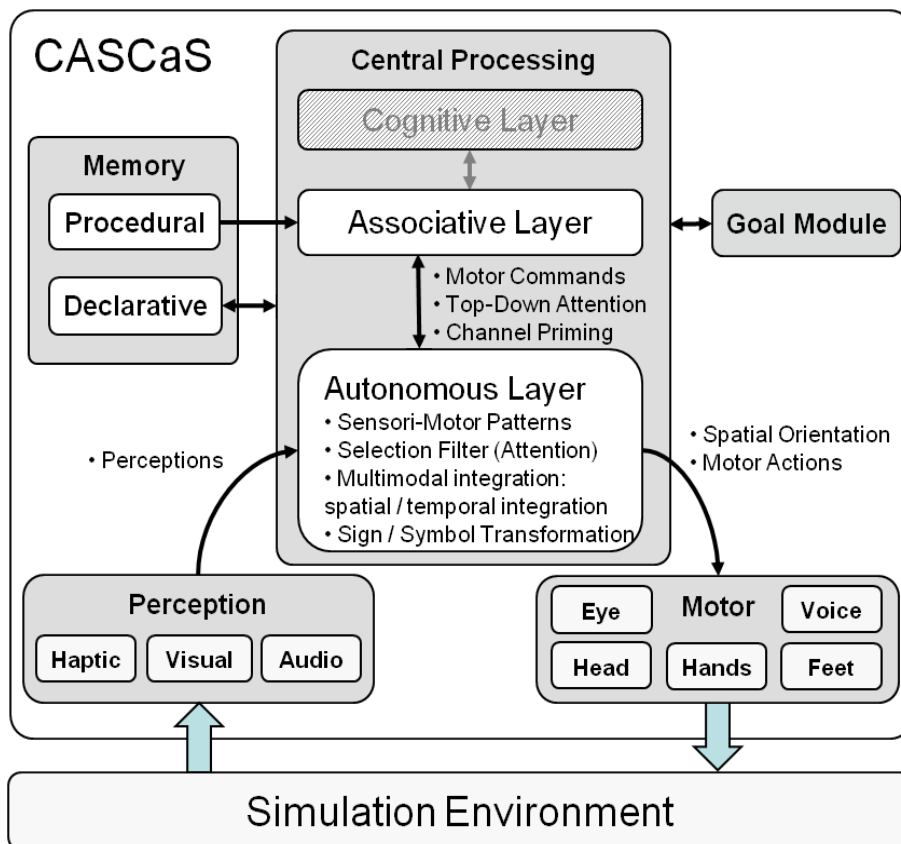


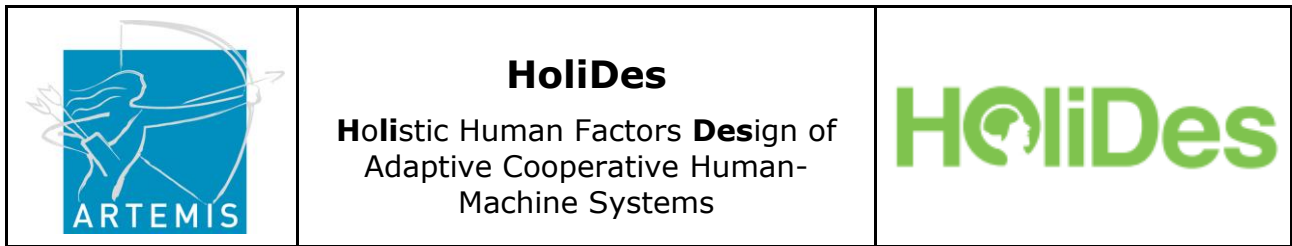
Regarding MOVIDA-AdCoS effectiveness, it could be possible for instance to replay several times the critical scenario previously presented in Figure 11, for testing different timings of the warning, in order to identify the last moment (in this scenario of reference) when human drivers are able to manage the risk from this type of warning, and when they are not able to avoid the collision even if warned, requiring in this case that the MOVIDA-AdCoS takes control.

## 2.1.2 CASCaS (OFFIS)

### 2.1.2.1 Description and Objectives

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

Figure 13 shows the current version of the architecture with all its components. Basically, the architecture consists of five components. The first one is the *Goal Module* which stores the intentions of the modelled human agent (what it wants to do next). Following, the *Central Processing* component is subdivided into three different layers: the cognitive layer, which can be used to model problem solving, the associative layer, which executes learned action plans, and the autonomous layer, which simulates highly learned behaviour. The *Memory* component is subdivided into a procedural (action plans) and a declarative knowledge (facts) part. The *Perception* component contains models about physiological characteristics of the visual, acoustic and haptic sensory organs, for example models about peripheral and foveal (central) vision. Finally, to interact with the external environment, the *Motor* component of CASCaS contains models for arm, hand and finger movements. It also comprises a calculation for combined eye / head movements that are needed to move the visual perception to a specific location.

In general, the model starts observing its environment via the Perception component and receives input which is stored in the Memory component. Depending on its current intention and on the perceived information from the environment, it selects action plans and tries to achieve its current goal. It may generate new goals and further actions, which can be triggered by events perceived from the environment or the model may itself create new goals, based on its own decision making process, to initiate a certain behaviour.

### **2.1.2.2 Application domain and Use Cases**

CASCaS will be applied in WP9 in cooperation with TAK for the evaluation of their User Interface/AdCoS. To this aim, CASCaS will be enhanced to calculate Saliency Maps for the user interface. Saliency Maps are a topographically arranged map that represents visual saliency of a corresponding visual scene. The implementation of this extension is in its final phase, and it is planned to be finished by month M30.

	<p><b>HoliDes</b></p> <p>Holistic Human Factors <b>Design</b> of Adaptive Cooperative Human- Machine Systems</p>	
--	--	---

### 2.1.2.3 Integration in the HF-RTP and interaction with other MTTs

The main purpose of CASCaS is the real time simulation. A dedicated framework for simulation (IEEE 1516 HLA Standard) has been chosen for interfacing CASCaS with other simulation tools. OSLC (Open Services for Lifecycle Collaboration) is not suitable for running real-time simulations, as OSLC does for example not integrate time management and synchronisation between clients. Therefore the use of OSLC for connecting simulation tools is not appropriate. Anyway, the surrounding tools for producing the needed input for CASCaS (e.g., MagicPED) or processing the output (e.g., Excel, Knime, and R) are candidates for integration with the HF-RTP as well as the Human Efficiency Evaluator (HEE, cf. D2.5). Indeed the HF-RTP integration of MagicPED, which is described in D2.5, has already been started.

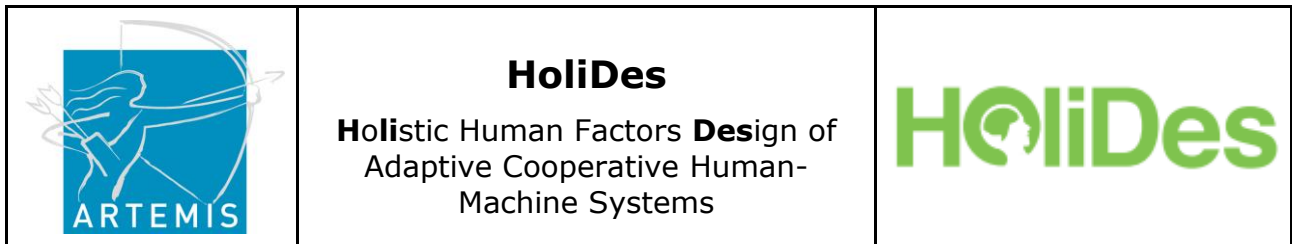
## 2.1.3 MDP-based Co-pilot (University of Torino)

### 2.1.3.1 Description and Objectives

The driver model developed by University of Torino for the CRF demonstrator in WP9 (also called co-pilot) has a central core which computes an “optimal manoeuvre” that is then suggested to the user through an appropriate, adaptive HMI. The modelling formalism used to realize the co-pilot is that of Markov Decision Process (MDP) (Puterman, 2005), a well-known formalism defined by Bellman in the early sixties for studying optimization problems.

An MDP is a stochastic control process that, at each time step, the modelled entity is in some state  $s \in S$ , and a decision maker may choose any action  $a \in A$  that is available while in  $s$ . Then, the process goes into a new state  $s'$  according to a specified transition probability (random choice), providing feedback to the decision maker in the form of a corresponding reward (or cost)  $R(a,s,s')$  (depending by the chosen action and by the source and destination state). A key notion for MDPs is the strategy, which defines the choice of action to be taken after any possible time step of the MDP. Analysis methods for MDPs can compute the strategies that maximize (or minimize) a target function based on the MDP's rewards (or costs). In this way the MDP model is used to compute the optimal manoeuvre, which is suggested to the human to achieve her/his goal.

The possible driving strategies of the co-pilot are planned using the informations of the AdCoS state, which are obtained from physical sensors.



Since the data of these sensors are subject to measurement errors and filtering, the MDP strategies are constructed using uncertain transition rates, i.e. rates are not a single value but are bounded by an interval. The decision process selects the strategy that behaves better in the worst case scenario, taking into account all the possible parameter variation induced by those AdCoS state parameters that are represented as intervals.



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

To cope with this aspect we use Markov Decision Petri Net (MDPN) (Beccuti M. F. G., 2007) a higher-level formalisms whose semantics is defined on an MDP.

MDPNs allow to specify the overall MDP behaviour as a composition of the behaviour of several components (some of which are subject to local non deterministic choice, and are thus called controllable, while the others are called non controllable). Another peculiar feature of MDPNs is that they allow any non-deterministic or probabilistic transition of the MDP to be obtained as the composition of a set of non-deterministic or probabilistic steps, each one involving a subset of components. A MDPN model is composed of two parts, both specified using the PN formalism with priorities associated with transitions: the  $PN^{nd}$  subnet and the  $PN^{pr}$  subnet, describing respectively the non-deterministic (ND) and probabilistic (PR) behaviour.

With respect to WP4 goals, the MDPN model of the co-pilot can be seen as a system to be verified against classical and specific properties, as well as against specific WP9 requirements. Classical properties can be, for example, reachability of states; specific properties can be the coherence between the set of strategies described by the MDPN and the task/subtask structure.

To account for uncertainty MDPN have been extended to MDPNU (MDPN with uncertainty) through a research collaboration with the University of Dortmund. An MDPNU can be defined through the GreatSPN graphical interface and automatically translated into BMDP (a class of MDP with uncertainty) and an optimal strategy is then found using the BMDP solver made available by the University of Dortmund. In MDPNU the uncertainty is introduced at the level of the transition rates in the  $PN^{pr}$  component (the probabilistic part of the MDPNU). A detailed description can be found in WP2 deliverable 2.5.

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

Moreover, we are studying new computational approaches to speed up the solution process of (B)MDP models. In particular we are investigating how to resort a) aggregation techniques to reduce the model state space, or b) approximate algorithms based on the well-known "finite-lookahead" techniques to obtain sub-optimal solution when time constraints severely limit the utilization of the classical solution.

### **2.1.3.2 Application domain and Use Cases**

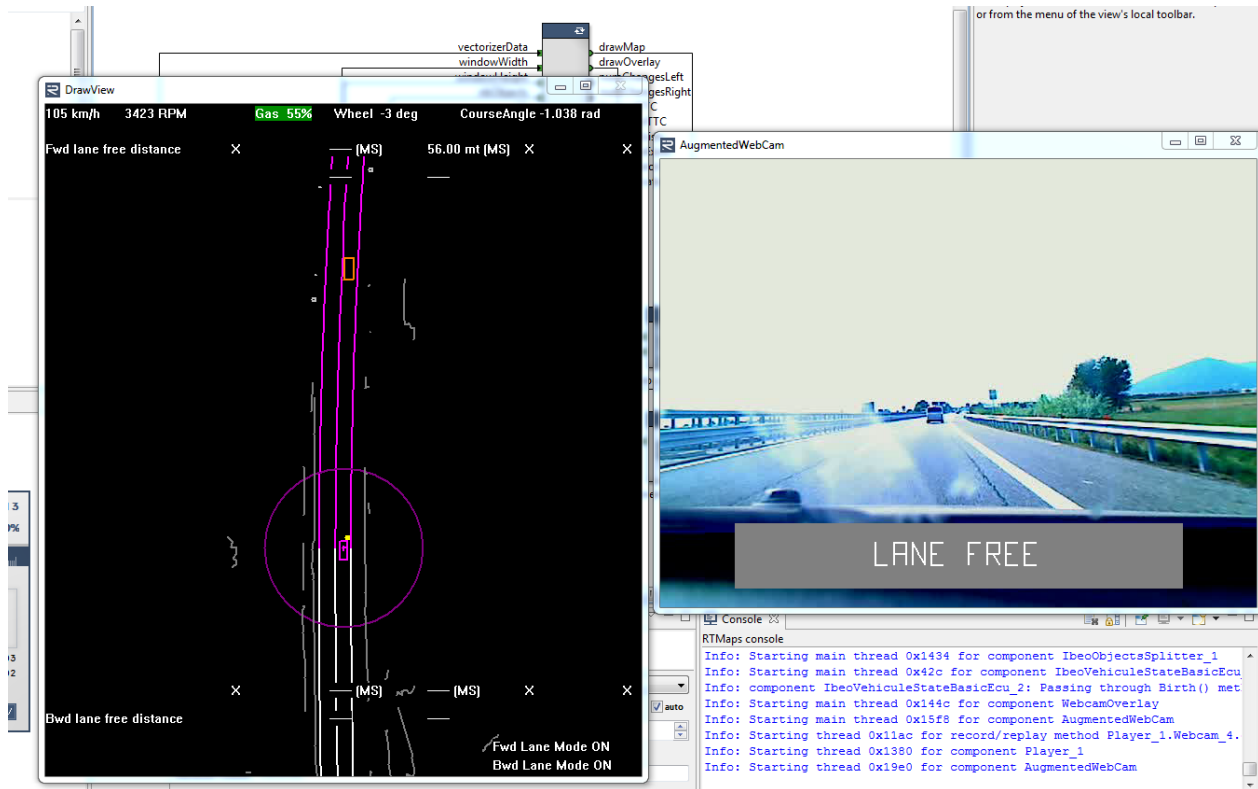
The MDP that realizes the driver model will be used in all the use cases that involve the CRF test vehicle (WP9). For the time being the attention is concentrated on the LCA (lane change assistance) use case, in particular UC4 of WP9. The architecture of the co-pilot to be run on the test vehicle is already illustrated in section 4.2.3 of deliverable D3.4. Adaptivity is accounted for by the MDP strategy: since the strategy (the MDP solution) is recomputed progressively, it may vary due to new changes in the AdCoS environment (internal – the state of the driver – or external – e.g. the traffic situation), providing different warning and intervention strategies (WISs) or suggestions on the driver dashboard, or on any other use of the Driver Model component output. When uncertainty is considered, the strategy can be chosen so as to optimize the worst, best or average case, with respect to the lower and upper bound of the transition (event) probabilities, and it will be particularly relevant in the context of WP9 to find strategies which are robust with respect to the transition bounds.



## HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

# HoliDes



**Figure 14: internal status of the co-pilot, with the output strategy.**

In particular we have currently defined the input information used to characterize the state of our (B)MDP.

Input data are taken from the following sources:

1. Driver status, represented by his distraction level and his driving intention.
2. Car status: the light indicators, brake pedal status, current speed and acceleration, and the turning angle of the wheel. These data are used to compose the MDP state.
3. Lane detected by the external camera. Lane detection provides the detected distance, the type of mark signs, the lane width and the car position in the lane. This is used to reconstruct the shape of the lanes, and infer the position of the car on the road, in order to design the MDP strategies for the lane change and car following scenarios.
4. External objects, provided by the LIDAR sensor. External objects are classified by the IBEO software stack as either fixed obstacles or moving obstacles. Detected objects are shown in Figure 14 as grey





lines for the fixed obstacles, and coloured bounding boxes for the moving objects.

The (sub)optimal strategy derived by solving the (B)MDP is instead provided to the adaptive HMI according to the specifications reported in Section 3.3 Deliverable 9.3.

Current prototype emits the reconstructed lane status, in terms of:

- Existence of left and right lanes, the possibility of crossing (depending on the lane marks) as well as the availability of a manoeuvre space in these lanes.
- Times-To-Collision with the incoming objects on the lanes, both forward and backward w.r.t. the vehicle position.
- A count of the number of lane changes

A synthesized output of the indication to be shown to the user is also provided in the prototype version, as shown on the right of Figure 14.

This indicator is modelled according to the rules defined in the task model of D9.3, and provides the single information that should be shown to the driver. This information is only shown for development purpose. In addition, there is no indication of the haptic or acoustic output.

### **2.1.3.3 Integration with HF-RTP and interaction with other MTTs**

The co-pilot core is an MDP solver, which is part of the GreatSPN tool, which has been enhanced for satisfying the Holidés requirements. For the time being the extension allows to account for parameter uncertainty, while, in the next months, it the solver could also be extended to include incremental strategy computation and approximate strategy computation (if the tests we are conducting will reveal that, when the solver will be on board, there is not enough time to re-compute a full strategy at each time step).

The MDP solver will be part of the run-time environment of RTMaps. For the time being, there has been no request from other partner to use the GreatSPN MDP solvers, so MDP solvers have been made available only as an RTMaps component or as a solution option of the GreatSPN tool.

Both GreatSPN and RTMaps will be used in the activity of model-based verification of the co-pilot, typical of WP4. In the following we describe how validation of MDP has been integrated in GreatSPN and how the MDP solver has been integrated into RTMaps.



## HoliDes

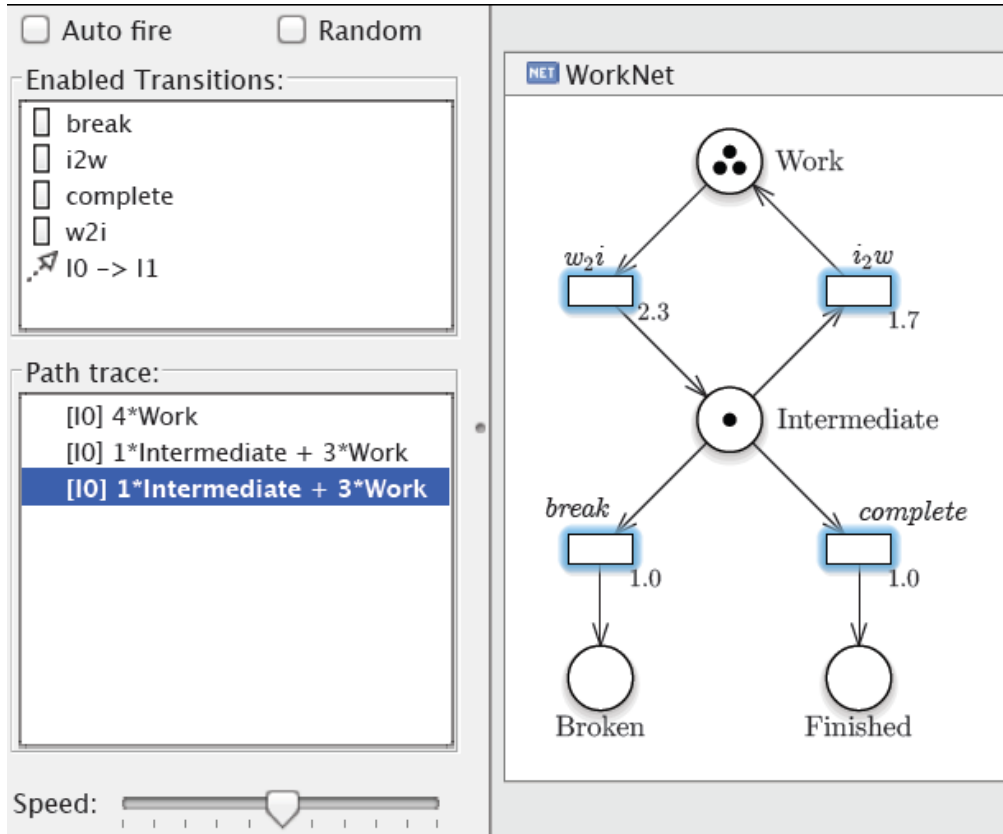
Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

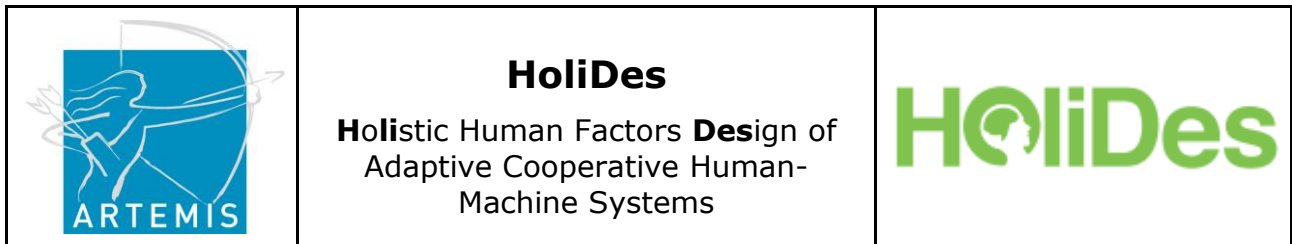
# HoliDes

### **Integration of MDP validation with GreatSPN.**

The GreatSPN suite of University of Torino provides a framework to design and solve MDPN models by means of specific modules [Beccuti M. et al (2011)]. The structure of the MDP solution in GreatSPN has already been described in D2.4, in this deliverable we want instead to describe the feature for MDP verification, which will be centred on the token game for MDPN and the model-checking of MDPN.

MDPN and token game. If the MDP is generated from an MDPN, as it is the case for the co-pilot model of WP9, then we can envision to extend the GreatSPN GUI to play the token game not only for classical and coloured Petri Nets, which is already available in the GUI, but to adapt it to MDPN: As a result the designer of the co-pilot will be able to animate the model and observe its evolution over time. Figure 15 shows the GreatSPN GUI while playing the token game: the possible events (transition firings) are highlighted (right box), while the boxes on the left allow observing the history of the animation.





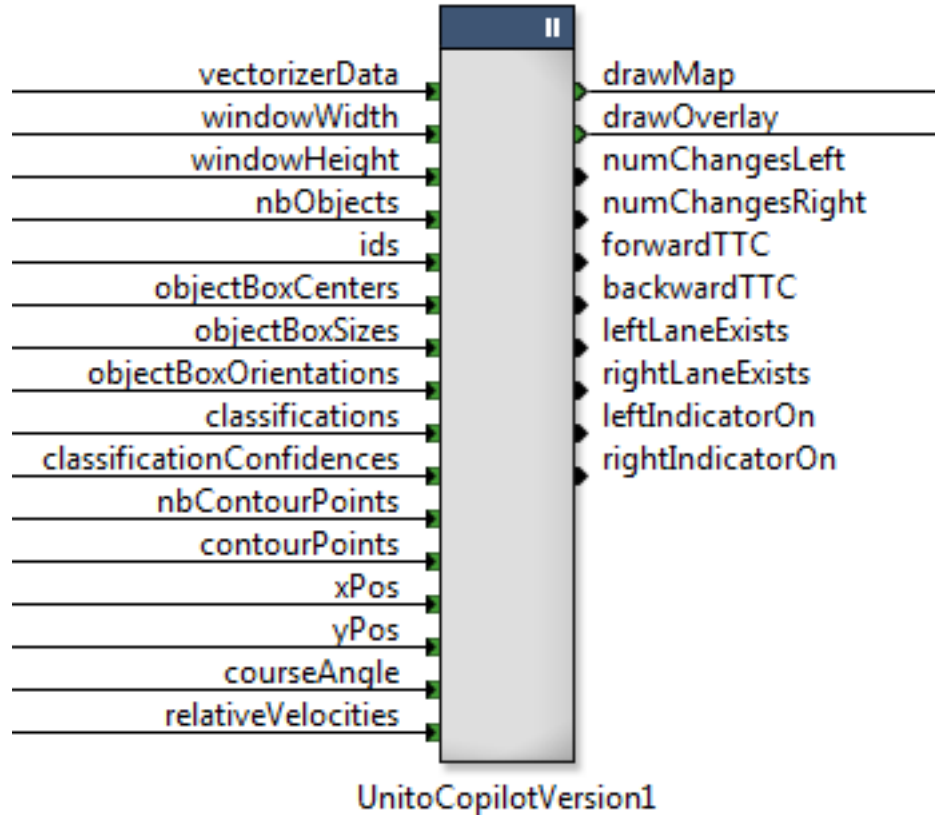
**Figure 15: GreatSPN GUI while playing the token game**

MDPN and model-checking. Model-checking is a state space exploration technique that allows checking properties defined in specific logics, like CTL or LTL, or their probabilistic/stochastic extensions.

GreatSPN includes a CTL model-checker, for qualitative verification, as well as a CSL/CSLTA one, for probabilistic verification of (coloured) Petri net models. There is no support for model-checking MDPN, but there is a functionality to translate the underlying MDP to an MDP in the format of PRISM (Kwiatkowska M., 2011), a well-known tool for probabilistic verification.

Moreover, we are working to enrich the GreatSPN suite with a new statistical model checker tool, in which Monte Carlo simulation approach is exploited to provide statistical evidence for the satisfaction or violation of the specification.



Of course, in contrast to a state-space approach, a simulation-based solution does not provide an exact result, but a result associated with a confidence level. However, simulation-based approaches are known to be far less memory and time intensive than numerical ones, and are sometimes the only option.



**Figure 16: The RTMaps component of the Co-pilot.**

***Integration of MDP solver into RTMaps.***

The RTMaps integration of the MDP co-pilot is realized by the implementation of a new component, shown graphically in Figure 16. This module implements the standard component life-cycle of RTMaps, including the input/output analysis, the synchronization with the multiple input sources, and the invocation of the internal GreatSPN solver. Solver output is streamed to the adaptive HMI, whose status is controlled by another RTMaps component (not shown in figure).

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

## **2.2 MTTs specifically dedicated to support AdCoS Modelling and their Virtual Simulation**

This section describes the techniques and tools that have been identified by the partners of HoliDes to specifically support the AdCoS design, modelling and verification.

### **2.2.1 RTMaps (INTEMPORA)**

#### **2.2.1.1 Description and Objectives**

The RTMaps software is a rapid and modular development environment for real-time applications handling multiple heterogeneous data streams. It has capability to support many data sources (such as video cameras, GPS, CAN bus, audio, motion capture, 3D sensors, DAQ, IMUs, laser scanners, radars, eye trackers and biometrics sensors, etc.)

RTMaps provides an accurate timestamp for each and every data sample entering the application and operates as a multi-threaded environment to be able to manage different data streams with different frame rates, including event-based sources.

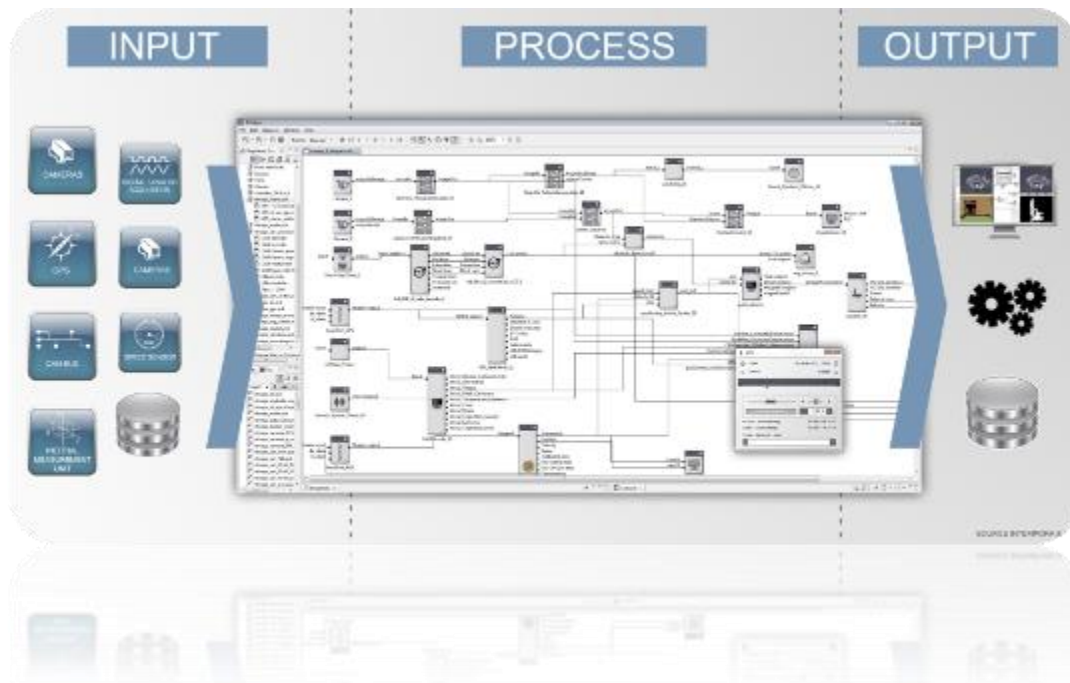
It is capable of recording and playing back any kind of data streams in a synchronized manner.



## HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

# HoliDes



**Figure 17: The RTMaps Studio**

RTMaps is particularly suited for the following applications:

- Perception thanks to multi-sensor data fusion
- Multimodal HMIs development

After Action Analysis of operator / driver / pilot behavior, it is included in distributed environment with cooperating operators.

### 2.2.1.2 Application domain and Use Cases

RTMaps is particularly suited for the following applications:

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

### 2.2.1.3 Integration in the HF-RTP and interaction with other MTTs

Since RTMaps is well suited for everything related to external perception and operator monitoring using multiple heterogeneous sensors, Intempora



## HoliDes

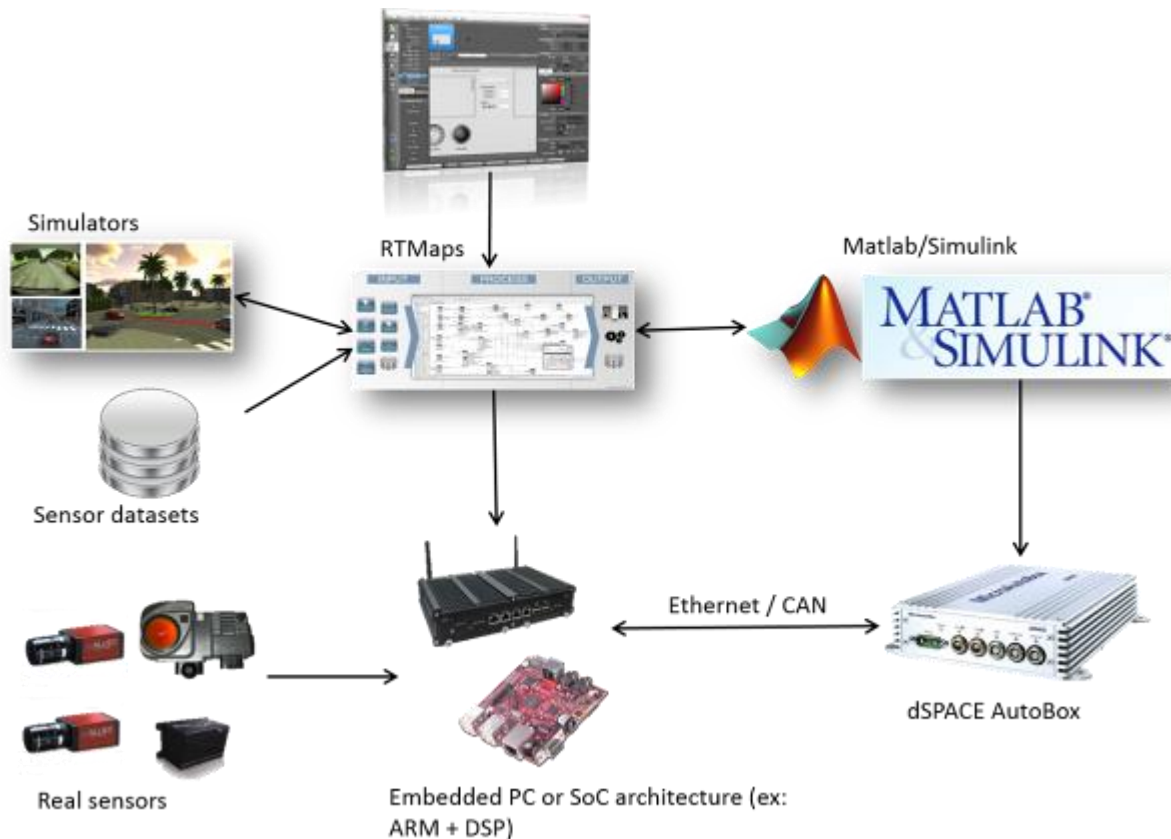
Holistic Human Factors **D**esign of  
Adaptive Cooperative Human-  
Machine Systems

# HoliDes

developed various interconnections with other complementary development environments in order to extend the capabilities of RTMaps and to propose a complete development toolchain for Adaptive Cooperative Systems taking in charge various sensors for perception and operator monitoring, command/control, and HMI (displays).

Among the interoperable tools are:

- Matlab and Simulink, mainly aimed at command-control laws development
- Qt/QML for graphical user interfaces
- In a similar approach as with Qt and QML for development of graphical user interfaces, an interface with djnn (see Section [2.3.2](#)) will be developed with ENAC.
- Simulators such as Pro-SIVIC (see Section [2.2.2](#)) for virtual testing



**Figure 18: An AdCoS development toolchain**



## HoliDes

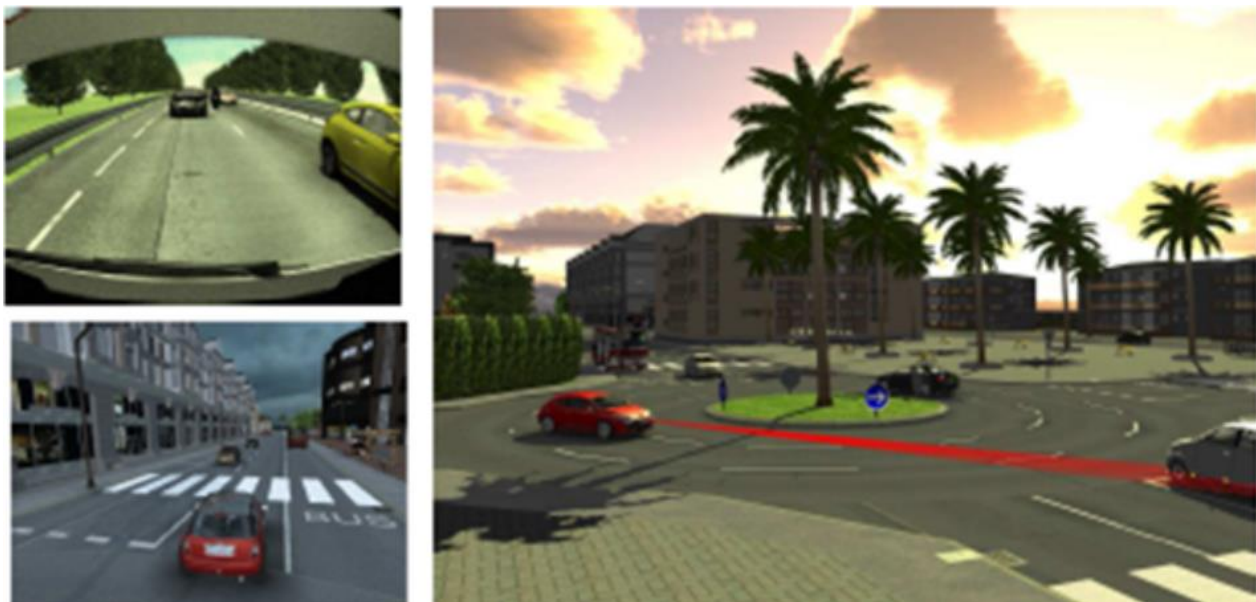
Holistic Human Factors Design of  
Adaptive Cooperative Human-  
Machine Systems

# HoliDes

### 2.2.2 Pro-SIVIC (CIVITEC)

#### 2.2.2.1 Description and Objectives

Pro-SIVIC<sup>®</sup> is software developed by CIVITEC. This tool is particularly suited for multi-sensor systems simulation in 3D environments. It allows simulating custom scenarios involving environment conditions, multiple sensors such as cameras, radars, laser scanners, IMUs, etc.





**Figure 19: The Pro-SIVIC<sup>®</sup> simulator**

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

Pro-SiVIC<sup>®</sup> allows simulating custom scenarios involving road environment conditions, multiple sensors, dynamic actors and people to perform prototyping and testing stages through simulation.



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

### 2.2.2.2 Application domain and Use Cases

Pro-SIVIC<sup>®</sup> is primarily dedicated to Automotive applications (demonstrations of previous work in this area are available at <http://www.civitec.com/applications/>).

It is a tool to support the development of Advanced Driver Assistance Systems (ADAS) and to simulate car sensors to support vehicle automation.

The software solution Pro-SiVIC<sup>®</sup> has been developed as an answer to the virtual design, prototyping and testing of such ADAS systems with a systematic approach thanks to powerful sensor simulation.

Pro-SiVIC<sup>®</sup> provides a software environment to simulate complex scenarios featuring multi-technology sensors while fully controlling the conditions of the test.

To accomplish this, Pro-SiVIC<sup>®</sup> helps:

- to compose scenarios
- to create and setup the actors in this scenario (vehicles, people, other dynamic objects)
- to define and configure the appropriate sensors
- to produce simulated data (store or exchange with another application)

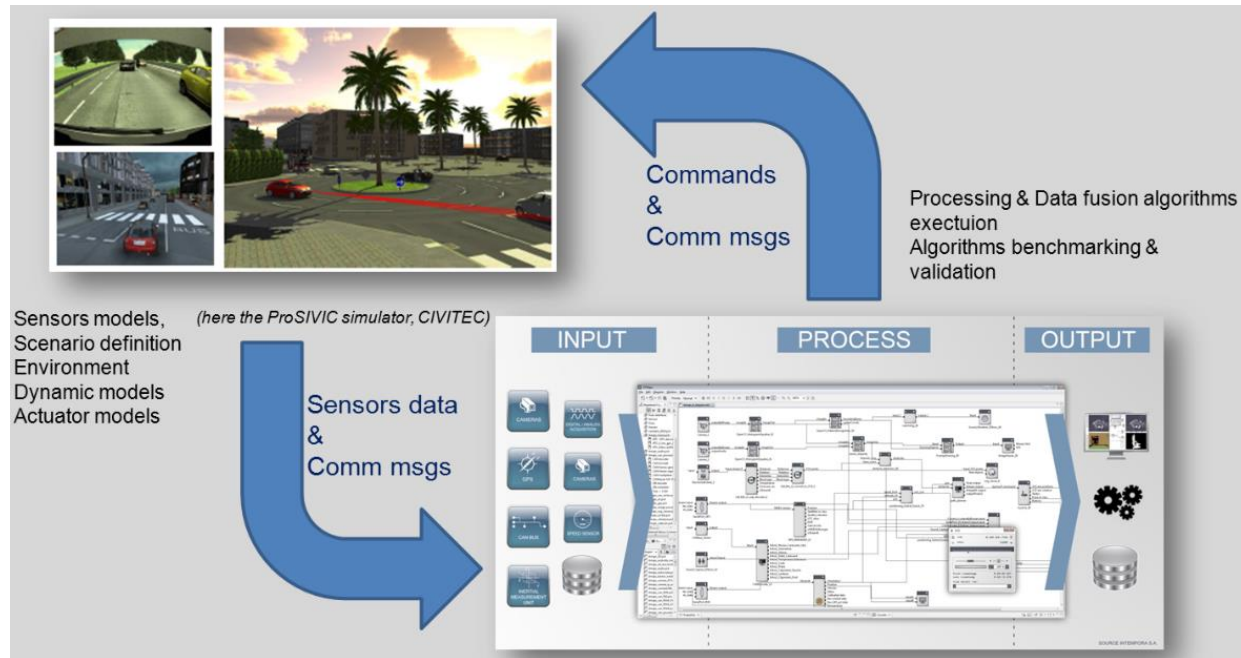
It then becomes possible to build and operate scenarios representative of real situations, complex and/or dangerous in dedicated climatic conditions (rain, fog, snow, brightness...).

In addition, simulation allows the evaluation of a large number of variants of scenarios, by allowing modifying the parameters influencing the behavior of the sensors, of the environment and of the mobile objects.

### 2.2.2.3 Integration in the HF-RTP and interaction with other MTTs



The various data streams generated by the Pro-SIVIC<sup>®</sup> simulator can be accessed from other software, via shared memory or network communications and using a dedicated API. However, regarding more specifically the HF-RTP of HoliDes, Pro-SIVIC<sup>®</sup> is already fully connected with

RTMaps (Fig. 9), and can be also connected with all other MTTs through this INTEMPORA software.



**Figure 20: Pro-SIVIC® interoperability with RTMaps**

Pro-SIVIC 3D simulator will provide in HoliDes accurate multi-frequency sensor models (cameras, lidars, IMUs, radars, etc.) as well as various environments and vehicle-dynamics models. Connecting Pro-SIVIC® virtual sensors and actuators to RTMaps is straightforward using on-the-shelf components establishing network or shared memory real-time communications. Using Pro-SIVIC® in conjunction with RTMaps will allow supporting portability of the virtually developed applications from desktop to the real prototype vehicles (by replacing the few components in charge of sensors and actuators interfaces).

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

## 2.3 MTTs for AdCoS design, verification and validation

### 2.3.1 ANaConDA, Race Detector & Healer, SearchBestie (Brno University of Technology)

These tools are provided by VeriFIT research group from Brno University of Technology (BUT) and are intended to support checking of concurrent software.

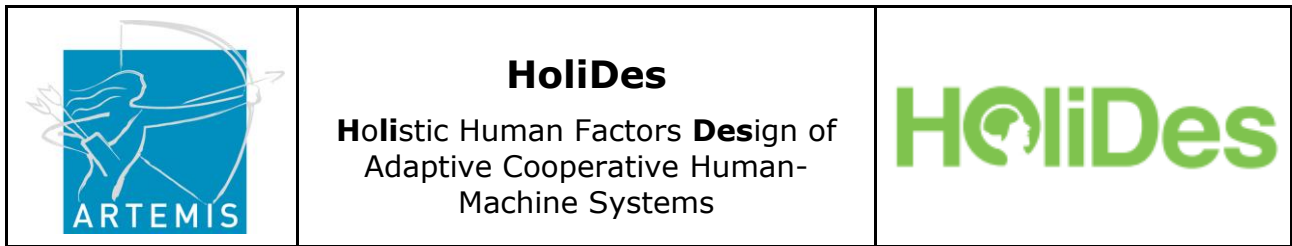
#### 2.3.1.1 Description and Objectives

**ANaConDA** (Adaptable NATive-code CONcurrency-focused Dynamic Analysis, available at <http://www.fit.vutbr.cz/research/groups/verifit/tools/anaconda/>) is a framework that simplifies the creation of dynamic analysers for analysing multi-threaded C/C++ programs on the binary level. The framework provides a monitoring layer offering notification about important events, such as thread synchronisation or memory accesses, so that developers of dynamic analysers can focus solely on writing the analysis code. ANaConDA also supports noise injection techniques to increase chances to find concurrency-related errors in testing runs. ANaConDA is built on top of the Intel's framework PIN for instrumenting binary code. Currently, it has been instantiated for programs using the pthread library as well as the Win32 API for dealing with threads.

ANaConDA framework aims at monitoring of primitives like function calls and/or memory accesses and, in a case of detected error, provides a user with back-trace to localise the error. The framework therefore fits for analysing programs written in, e.g. C/C++.

Since the framework itself depends on Intel's PIN framework, it fully supports only binaries with Intel-compatible instructions. ANaConDA framework consists of several components:

- Intel's PIN Framework needed for code injection of monitored program under test,
- one or more program analysers,
- ANaConDA core which controls the injections and call-backs to analysers.



Since ANaConDA is implemented as a *pintool* (a plug-in for the PIN framework), the framework provides several useful run-time pieces of information about:

- memory access (reads, writes, and atomic updates),
- synchronization (lock acquisitions/releases, signalling conditions),
- thread execution control (start and finish),
- exception (throws and catches), and
- back-traces (function return addresses).

An *analyser* of ANaConDA focuses on a specific property of program under test (for instance, monitoring memory accesses or lock operations). Analysis provided in a way of plug-in and communicates with ANaConDA core via pre-defined call-backs. An example of a simple analyser which monitors lock operations is in Figure 21.

```
PLUGIN INIT FUNCTION()
{
    // Register a callback function called before a lock is released
    SYNC BeforeLockRelease(beforeLockRelease);

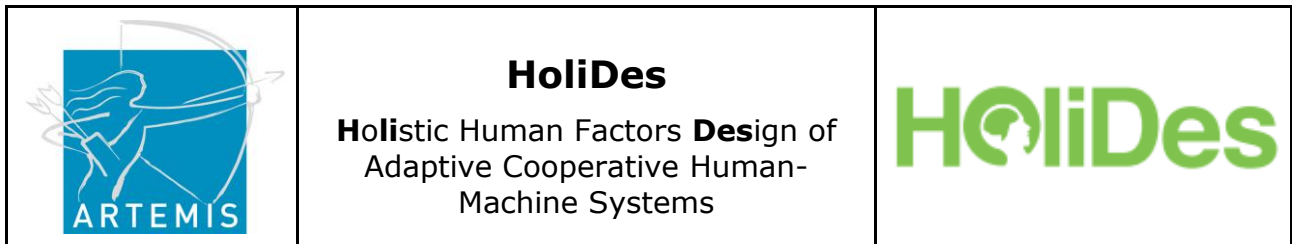
    // Register a callback function called after a lock is acquired
    SYNC AfterLockAcquire(afterLockAcquire);
}

VOID beforeLockRelease(THREADID tid, LOCK lock)
{
    CONSOLE("Before lock released: thread " + decstr(tid) + ", lock " +
            lock + "\n");
}

VOID afterLockAcquire(THREADID tid, LOCK lock)
{
    CONSOLE("After lock acquired: thread " + decstr(tid) + ", lock " +
            lock + "\n");
}
```

**Figure 21: A simple analyser monitoring lock operations**

Considering the use of ANaConDA for analysis of concurrent programs (and search for concurrent bugs), the framework also supports fine-grained combinations of noise. A noise is external influence of the thread scheduler (i.e. context-switch timing). Noise injection therefore attempts to increase the chances to see the rare executions leading to an error. ANaConDA



supports specification of noise injection that might be used for memory accesses or lock operations (cf. Figure 22).

```
[noise]                # Global noise settings
type = yield           # Insert calls to yield
frequency = 100        # Inject noise in 10% of times
strength = 4           # Give up the CPU 4 times
[noise.read]          # Noise settings for read accesses
type = yield           # Insert calls to yield
frequency = 200        # Inject noise in 20% of times
strength = 8           # Give up the CPU 8 times
[noise.write]         # Noise settings for write accesses
type = sleep           # Insert calls to sleep
frequency = 400        # Inject noise in 40% of times
strength = 2           # Sleep for 2 milliseconds
```

**Figure 22: An example of different noise settings for reads and writes.**

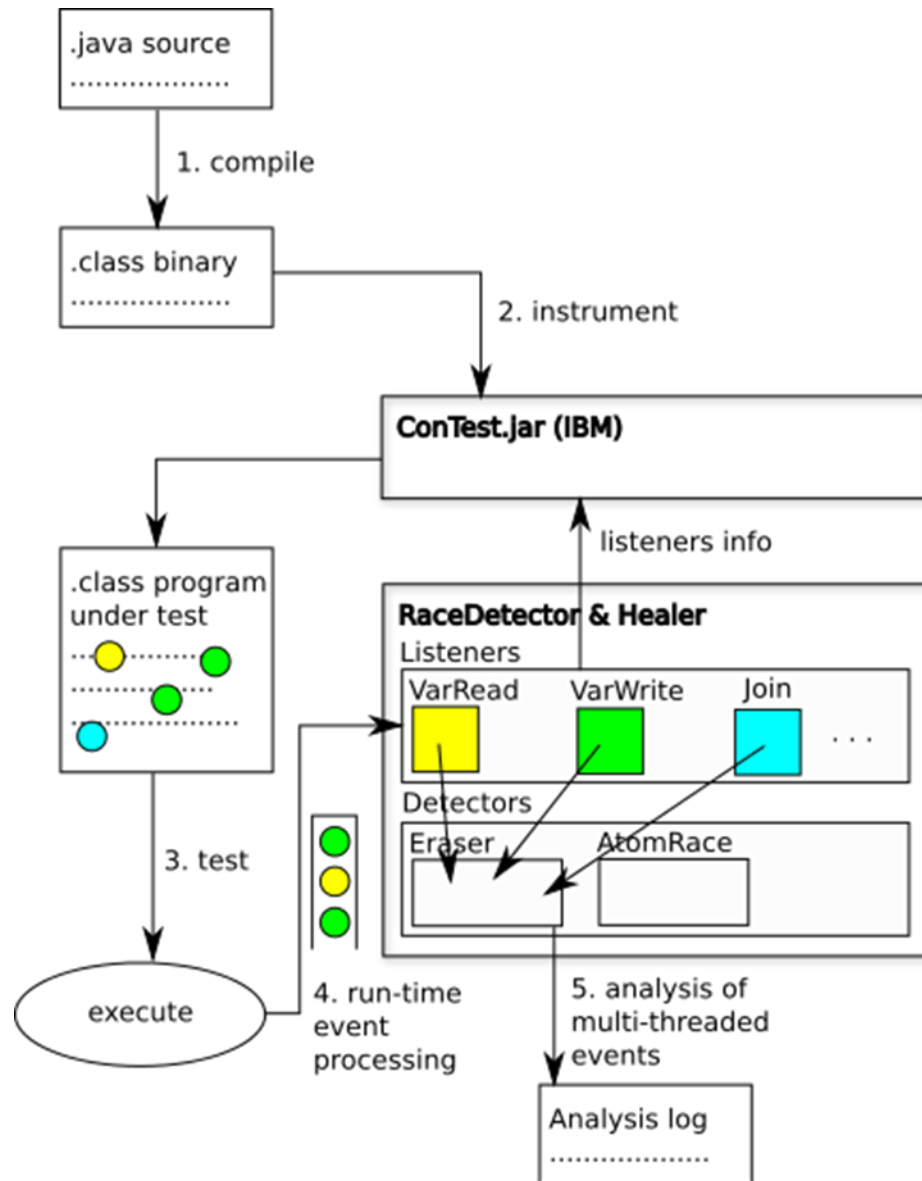
The **Race Detector & Healer** (available at <http://www.fit.vutbr.cz/research/groups/verifit/tools/racedetect/>) is a prototype for run-time detection and healing of data races and atomicity violations in concurrent Java programs. The tool uses the IBM ConTest listeners' architecture for tracking the program behaviour and analysing it.

The Race Detector & Healer tool can use either a modified version of the Eraser algorithm to detect violations in a locking policy or the AtomRace algorithm for detecting atomicity violations. The tool identifies locks or atomic sections when accessing a shared variable. Detection of such events is done by instrumenting Java binaries (.class files) via IBM's ConTest framework (cf. Figure 23, step 2). Simultaneous access, in particular with write access, implies possible data race. Of course, a chance on detecting such situation is sometimes very low. Therefore this approach can be combined with noise injection technique which injects noise near accesses to shared variables, thus makes them increase the probability of detecting a data race. The basic idea of healing is that the Race Detector & Healer tool tries to force the predefined correct atomicity. If there is a race or atomicity violation over a variable and if there is predefined atomicity present, the Race Detector & Healer tool use selected method to minimize the probability of context switch in the middle of the problematic atomicity section.



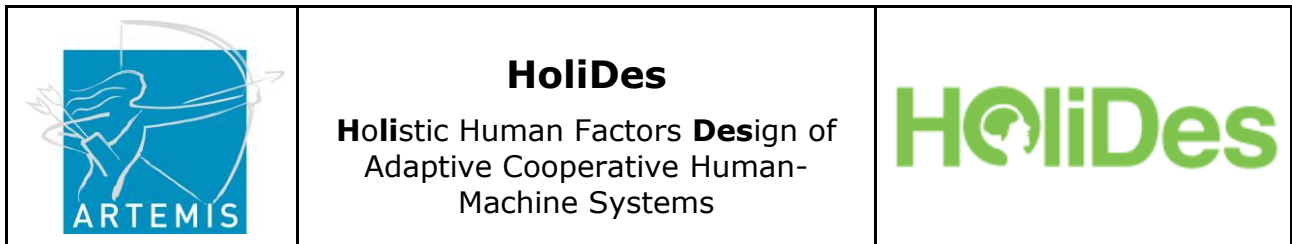
# HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems



**Figure 23: A scheme of usage Race Detector & Healer when analysing concurrent Java programs.**

**SearchBestie** (Search-Based Testing Environment, available at <http://www.fit.vutbr.cz/~iletko/wiki/doku.php?id=searchbestie:main>) is a generic infrastructure that is designed to provide environment for experimenting with applying search techniques in the field of program testing



(e.g., to find optimal settings of injected noise to increase efficiency of ANaConDA and Race Detector & Healer).

### **2.3.1.2 Application domain and Use Cases**

The tools aim mainly at detecting concurrent errors which fit to systems where many machines cooperate together. As a consequence, it is possible that the tools may be used for either (i) for AdCoS design Verification, or (ii) Verification of requirements which deal with assurance of availability of an object, data integrity, and responsiveness of a system.



More specifically, possible properties interesting for checking by the tools when fulfilling the requirements are, for instance (cf. D4.1), assurance that failure is spotted and adaptation is performed correctly [WP7\_HON\_AER\_REQ45], system responsiveness / adaptation to internal workload [WP7\_HON\_AER\_REQ46], or Verification of data integrity [WP9\_IFS\_AUT\_REQ03]. We have installed existing version of ANaConDA in Honeywell and discussed their requirements on environments used for the AdCoS. Based on that we have developed during the last months a new version of ANaConDA that is able to exploit current version of PIN framework under Windows 64-bit operating systems.

There is an interest in using Race Detector & Healer tool within WP6 – Health use cases expressed recently by Carlos Cavero Barca and Miriam Quintero Padron from ATOS.

### **2.3.1.3 Integration in the HF-RTP and interaction with other MTTs**

ANaConDa framework and its analysers focus on the testing phase of a project. Since OSLC is centered on development phase and is not optimized for the run-time/real-time data, implementing OSLC adaptor would sacrifice the effectiveness of the framework. There are no plans and Honeywell requirements to do that yet.

Race Detector & Healer works only in conjunction with instrumented Java programs and IBM ConTest which calls Race Detector & Healer tool at runtime. We have realized that there is an obstacle with application of Race Detector & Healer due to license issues connected with IBM ConTest (<https://www.research.ibm.com/haifa/projects/verification/contest/>). Despite there were some promises from IBM Research Labs Haifa (we have

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

cooperated with the team that developed IBM ConTest within some previous projects) that IBM ConTest will be released for use without such licensing, it has not happened so far. We have therefore decided to replace IBM ConTest by RoadRunner (<http://dept.cs.williams.edu/~freund/rr/>).

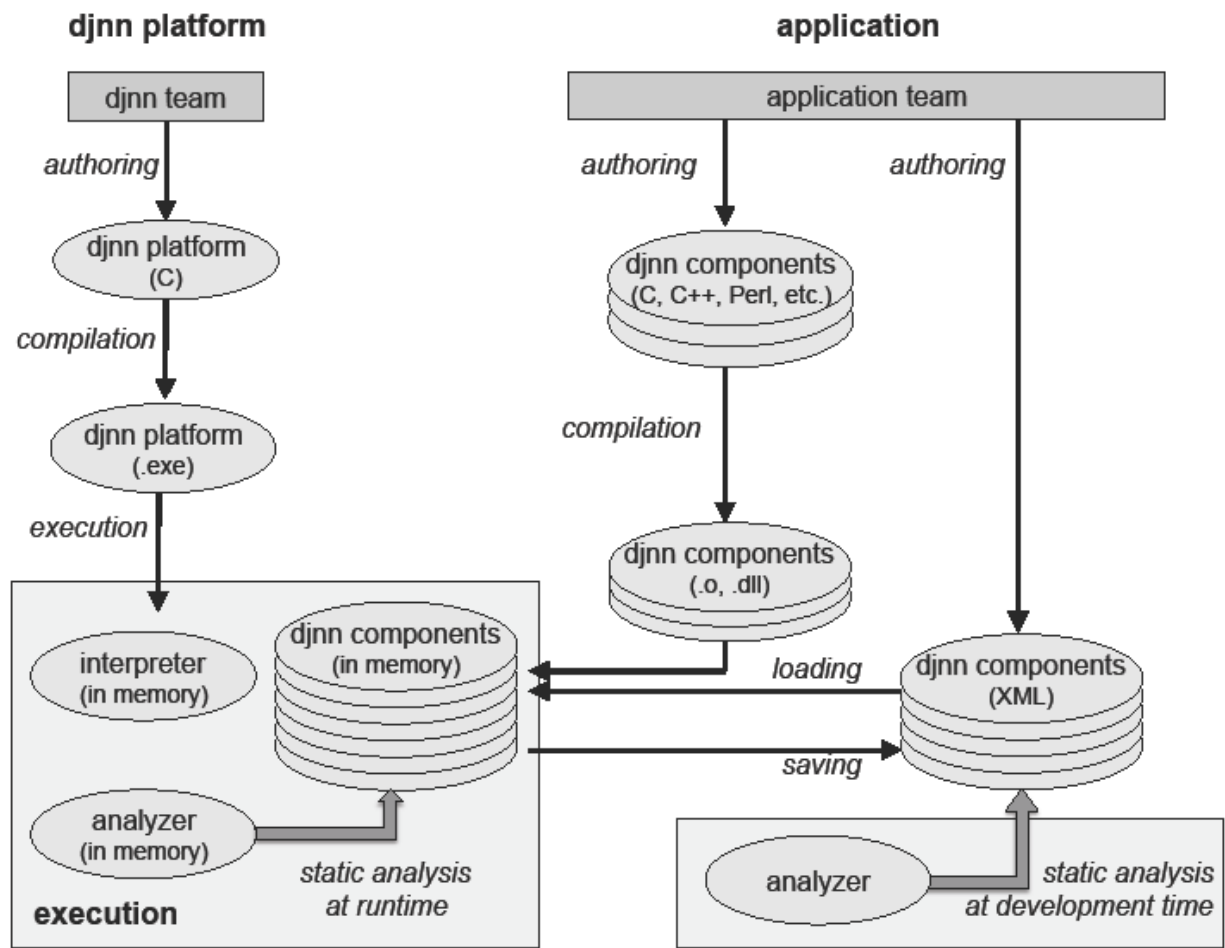
## 2.3.2 Djnn (ENAC)

### 2.3.2.1 Description and Objectives

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 an 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.





**Figure 24: djnn platform architecture**



djnn extensions

For the purpose of HoliDes WP4 (Verification of AdCoS), djnn framework is extended according to two different ways: static analysis and dynamic analysis.

- **Extension 1: djnn for static analysis**

Through this approach, djnn components are used to support verification of various properties at design time. This is achieved by two methods:

- **Extension 1.1: Static analysis of djnn models:** properties are checked on djnn model.

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

The djnn architecture, that separates an interpreter and an executable set of components, provides two solutions for performing static analysis of the components. The hierarchy of components can be analysed in memory, whatever method was used for creating it. This can be done by adding analysis tools to the djnn interpreter. Alternatively, analysis can be performed on XML files using dedicated analytical tools.

- **Extension 1.2: External analysis:** djnn components and djnn platform are translated to an external model on which verification is performed.

In HoliDes context, we selected Petri nets as an external target model supported by the GreatSPN tool.

- **Extension 2: djnn for dynamic analysis**

djnn executables are used to support the verification of properties at runtime, during the AdCoS simulation. For this purpose, two methods are proposed:

- **Extension 2.1: integration of djnn into a simulation platform.**

Djnn components are included in a platform dedicated to the simulation of AdCoS.

- **Extension 2.2: new mechanisms for verification at runtime.**

Dedicated mechanisms are also added to djnn to support verification at runtime.



These extensions are currently being developed. We provide hereafter the current situation.

### **Detail of extension 1.1: Static analysis of djnn models**

Remark: this part has been subject of an official communication. Refer to [Chatty 2015] for more information.

#### djnn applications: a tree of 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.

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

Programmers create interactive programs by instantiating and assembling software components, and connecting them to hardware components. The djnn environment provides them with 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.

djnn provides the control structures that have been introduced for interactive software in the last decades, as well as traditional computation-oriented control structures. This includes:



- bindings, which ensure simple reactions to events: when two components are interconnected by a binding, activation of the first triggers activation of the second;
- connectors, which ensure that any modification of their input value are propagated to their output values;
- state machines, whose transitions can be triggered by the activation of components, and whose states and transitions can trigger the activation of other components;
- 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;
- switches, which activate one among several components depending on the value of their state.

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.

djnn provide means to name and to organize components. The hierarchical structure allows programmers to think of their applications as trees of components, to address components as tree branches, and to reuse whole branches as components in other applications.

#### Using the tree components for verification

The hierarchy of components in djnn carries a significant part of the semantics of a djnn program. Consequently, a number of properties can be analysed by performing pattern matching in the tree.

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

### Pattern matching with XPath



XPath is a language specified by the WWW Consortium [W3C 2010], originally dedicated to the exploration of XML trees. It defines a rich grammar to build expressions that can be used to select nodes in a hierarchical document. The syntax of an XPath expression can be quite complex. It offers various constructs to specify a search path (relative, absolute), to express relationships, called axes, between the nodes (parent, children, sibling, descendant, etc.) and to check properties of the elements or their attributes through logical and arithmetic expressions. Processing an XPath query results in a list of items matching the expression. For example the expression `expr= (/widget/descendant::*/button)` retrieves all the "button" elements that descend from the "widget" element. Although XPath was designed for XML, it can be used with any hierarchical structure that is similar enough to the ontology of an XML document: element, element value, attribute, attribute value, etc. This is the case of the hierarchies of components in djnn, both in their XML form and when loaded in the djnn interpreter. When executing a djnn program, two phases can be distinguished: the building of the hierarchy of components in memory, then its execution. We focus on the first phase, in which djnn can perform static analysis automatically before running the program. This provides a convenient time, if not always optimal, to provide developers with feedback on the robustness of their code.

### Verifying component interface

Interface of a component is made of information the component is able to offer to other components: size, location of a mouse click, etc....

Most modern developments involve teams of developers that build or evolve components in parallel and then combine them. In the case of graphical user interfaces, this concurrent engineering can even involve different professions, with graphical designers producing the visual components and programmers producing the rest. The efficiency of this development process would be improved if all developers could verify before integration that their components follow the contract that was originally decided upon. Here, a number of such contracts can be defined as the existence of a specific pattern of descendants in a component.

Suppose for example that one creates a new widget for a WIMP toolkit. It must then be checked that this widget has a width and a height children to

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

ensure that it will be possible to connect them to the layout system. Such a verification can be made by checking that the XPath expressions `expr= (/widget/width)` and `expr= (/widget/height)` do not return a null result. Similarly, before adding a drag and drop behaviour to a component, we must check the existence of the `press/x` and `press/y` patterns, that is the event and properties that will trigger the behaviour, as well as the existence of and `x` and `y` children, that is the properties that will be changed by the behaviour. The corresponding XPath requests are, respectively, `expr= (/component/press/x)`, `expr= (/component/press/y)`, `expr= (/component/x)` and `expr= (/component/y)`.

#### Verifying execution context



The specification of an interactive system sometimes includes the expression of constraints on the hardware and software environment in which a given application will run. For instance, some applications require a minimum screen size to ensure readability and others require a specific input device to support a specific interaction style. Some of these constraints can be checked through XPath requests over the extended djnn tree, which includes the execution context.

The djnn framework gives programmers the possibility to explore an extended component hierarchy that represents the context in which their program runs. Their own components, when created, become part of this extended tree. Upon request, the extended tree can contain the available physical displays, the input devices, the batteries, etc. Once initialized, the extended tree can be explored like the application tree and its components can be used in control structures in the application tree. Consequently, verifying that a large enough physical display is available amounts to verify the existence of a specific component in the display tree through an XPath request such as: `(/displays/display[@width > 800])`.

#### Verifying component visibility

As mentioned earlier, the order of graphical components in the tree determines what will effectively be displayed. Based on this, it becomes possible to verify some properties such as the visibility of a graphical component.

For example, a property like “a component C must always be visible” can be checked with the following steps:

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

- The component C is the rightmost in the tree (so that no other component can overwrite it)
- No opacity component is located on its left (so that no component can affect opacity of C)
- If an opacity component is located on its left, its value is low (so that the component cannot affect the visibility of C) and no other component can modify the value.

#### Verifying the control flow

The combination of XPath queries to select elements and simple algorithms over their results allows addressing related to the control flow. These verifications are currently being studied.

#### **Extension 1.2: Translation of djnn model to Petri net model**

Remark: this part has been subject of an official communication. Refer to [Prun 2015] for more information.

Semantic of djnn model is expressed through Petri Nets [Jensen 1996] extended with reset arcs [Dufourd 1998]. We chose this formalism because, at a first glance, it offers good characteristics to represent both static and dynamic concerns through a state-transition semantic.

All djnn components are currently being individually modelled with Petri Nets. We provide here 2 examples: binding and assignment.



# HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

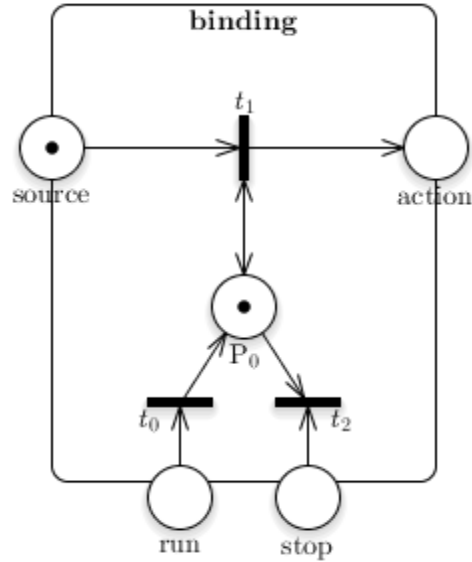


Figure 25: djnn binding component expressed in Petri net

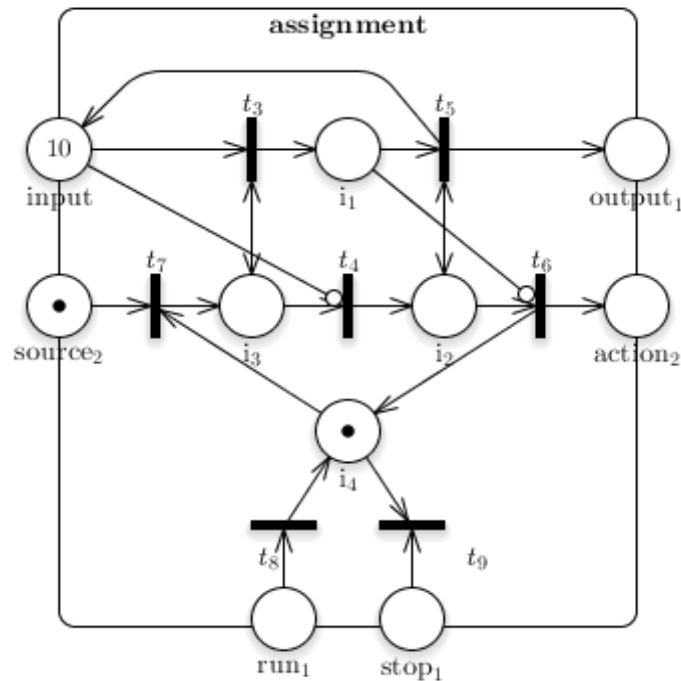


Figure 26: djnn assignment component expressed in Petri net



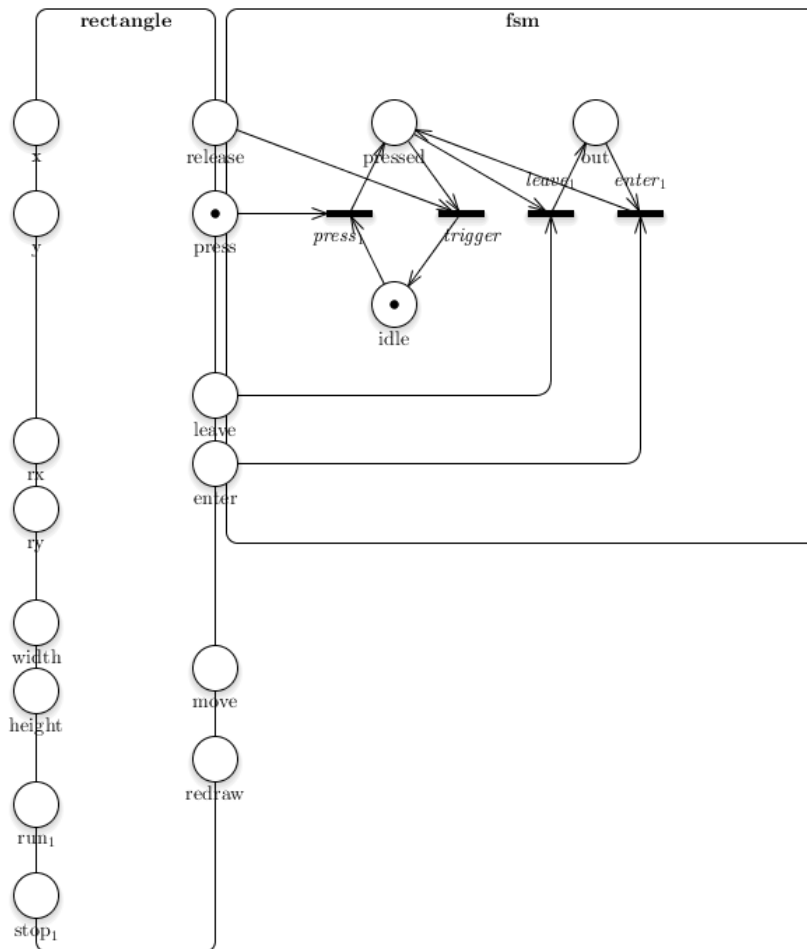
## HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems



Composition of components is expressed asynchronously through place merging.

Example: Figure 27 shows 2 components which are connected together. The first component is a rectangle modelled through various properties expressed with dedicated places (position X and Y, dimensions width and height, mouse event management release, press, enter and leave). The second component is a state machine made of 3 states (pressed, out and idle) and 4 transitions (press, trigger, leave and enter). Transitions of the state machine are triggered by properties showed by the rectangle. For example when the mouse is pressed on the area of the rectangle and the state machine is in idle state, and then the state machine jumps to pressed event.





**Figure 27: Approach for verification**

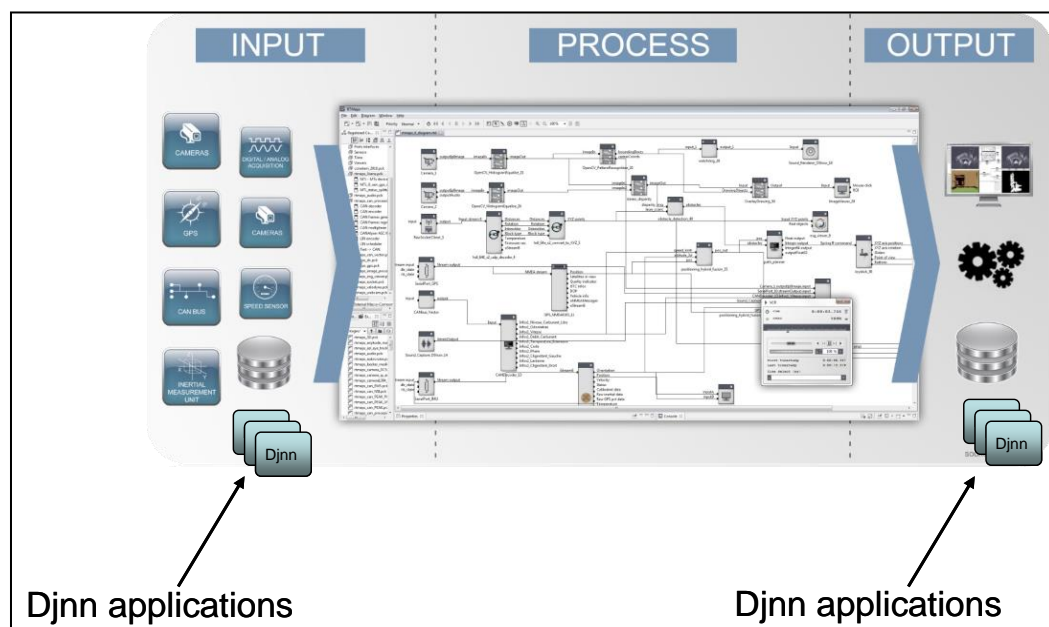
The verification of a property on djnn application is performed as follow:

- djnn application is translated to petri net model
- property is checked directly on the Petri net model. Reachability, Liveness and Boundedness are classical properties that can be checked on Petri nets.

### Details of extension 2.1: integration of djnn into a simulation platform

With this approach djnn components are plugged into a dedicated environment where they are executed for dynamic evaluation.

The environment should provide interfaces with other components of the AdCoS and, if necessary, with simulated components (inputs/output devices, for example). The environment should also support for interfaces with AdCoS external environment. The environment should also provide ways to support dynamic analysis with record/replay mechanisms and possibilities to modify the AdCoS environment.



According to this approach of verification, djnn has been interconnected with RTMAPS through a communication layer: DDS.



## HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

# HoliDes

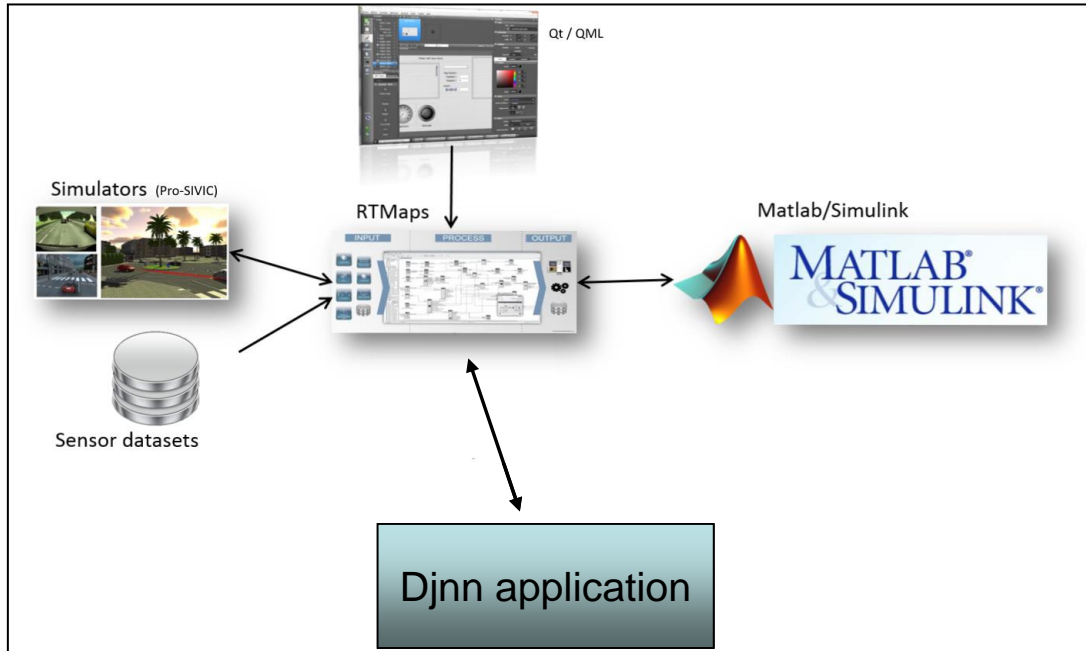




Figure 28: djnn RTMaps integration

### Introduction to DDS

Data Distribution Service (DDS) is a standard from Object Management Group (OMG).that aims to enable data exchange between computers [OMG 2015].

DDS is a specification of a middleware that handles complex network programming. It relies on a publisher/subscriber model for sending and receiving data, events, and commands among the nodes. Nodes that produce information (publishers) create "topics" (e.g., temperature, location, pressure) and publish "samples". DDS delivers the samples to subscribers that declare an interest in that topic. Any node can be a publisher, subscriber, or both simultaneously. Subscribers can be on different platforms from the publishers.

DDS provides several services that simplify the programming. The applications never need information about the other participating applications, including their existence or locations. DDS transparently handles message delivery without requiring intervention from the user

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

applications, including: determining who should receive the messages, where recipients are located and what happens if messages cannot be delivered

Today, there are several implementations of DDS protocol. Main commercial ones are RTI Data Distribution Service and PrismTech OpenSplice. An open implementations is also available: OpenDDS. Several Application Programming Interfaces (APIs) are available such as ADA, C, C++, C#, Java, Scala, Lua, Pharo and Ruby. Different Operating System (Linux, Windows) are supported. DDS specification ensures total interoperability whatever the implementation, the APIs and the OS are.

### **Setting up the connection**

#### Main principle

The main object shared across the network is called a topic. A topic is defined by three elements:

- name: identify the topic within the domain
- type: the programming type of the topic. Like a C/C++ structure
- QoS: a policy that express the non-functional properties of this topic (reliability, history, persistence)

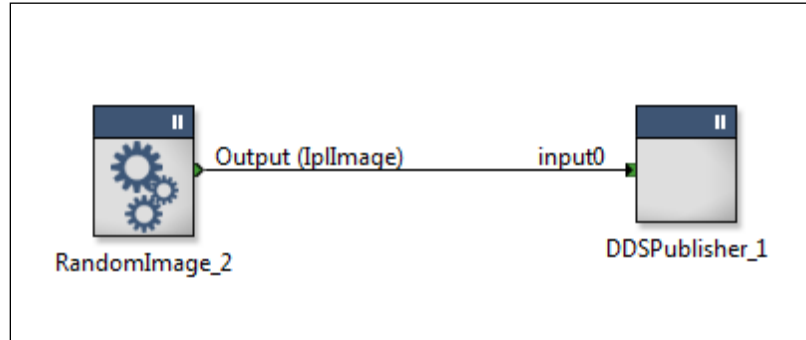
A topic instance is called a sample. Those samples are shared through subscribers and publishers. A publisher uses a data writer to produces samples, while subscribers use data reader to access to samples.

To talk to each other's publishers and subscribers must use the same topic, i.e. same name, same type and a compatible QoS.

#### Connection between RTMAPS and Djnn

RTMAPS is shipped with an OpenSpliceDDS version. It is installed in the packages/RTMaps\_dds directory of RTMaps installation (the RTMaps package RTMaps\_dds.pck is located there). Two components can then be used to support communication through DDS:

- DDSPublisher: used to send data through DDS
- DDSSubscriber: used to receive data through DDS



**Figure 29 Example of a DDSPublisher component.**

The DDSPublisher has the following properties:

- Topic\_name : this is the topic name (see before)
- Durability: this is a QoS parameter. This could be:
  - Volatile : No need to keep data instances for late joining data readers
  - Transient local : Data instance availability for late joining data reader is tied to the data writer availability
  - Transient : Data instance availability outlives the data writer
  - Persistent : Data instance availability outlives system restarts
- History: this QoS can be set to
  - Keep All: The History QoS can be set so keep all samples produced by the writer and not yet taken, until resource limits are not reached
  - Keep Last K: The History QoS can be set so to always have the latest K samples

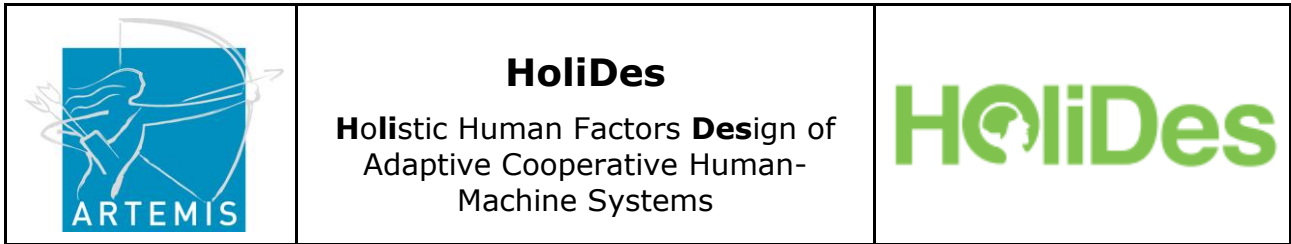
A type for the DDSPublisher is automatically chosen by RTMaps when the component receives its first data. Available types are: Integer32, Integer64, Float32, Float64, Stream8, Text, IplImage, MAPSImage, CANFrame and RealObject.

Djnn has a specific component to enable communication through DDS:

- RTMaps: used to send and/or receive data through DDS.

This component has the following properties;

- 2 children: in and out.
- A topic name for reception
- A topic name for emission



There is only one type which is handled: Text. That means that data with different types must be translated into text before being sent. Conversely, data received from djnn must be in text format.

### **Reference manual**

The djnn reference manual has been updated in order to add the description of the component RTMaps access:

RTMaps access

Description

RTMaps is a component that allows to send and receive data with RTMAPS.

Constructor

```
djnElement* djnCreateRTMapsAccess (djnnElement* parent, const char* name, const char* rname, const char* ename)
```

Create a RTMaps access element named `name`. The element will receive data according to `rname` topic. The element will send data according to `ename` topic.

Children

`in`: text property whose value is set whenever a text is received from RTMaps.

`out`: text property whose value is sent to RTMaps whenever it is set.

Standard actions

`RUN`: activates the element

`STOP`: stops the element

### **Example:**

Code below is a very simple example showing how an RTMaps component can be used.

When a piece of data is received from RTMaps, it is immediately sent to a text printer component whose function consists in displaying data on the screen.



## HoliDes

Holistic Human Factors Design of  
Adaptive Cooperative Human-  
Machine Systems

HoliDes

```
#include <djnn/core.h>
#include <djnn/base.h>
#include <djnn/gui.h>
#include "djnn/RTMaps.h"

int main (int argc, const char** argv)
{
    /* various initialisations */
    djnInitCore(); djnInitGUI(); djnInitBase(); djnInitRTMaps();

    /* declaration of components */
    djnElement *c, *a, *t;

    /* create the root component */
    c=djnCreateComponent (0, "root");

    /* create a RTMaps access component */
    /* rtmeps0 is the name of the topic used for reception */
    /* rtmeps1 is the name of the topic used for emission */
    a=djnCreateRTMapsAccess(c, "access", "rtmaps0","rtmaps1");

    /* create a text writer component */
    t=djnCreateTextPrinter (c, "text");

    /* create a connection between data reception from RTMaps and */
    /* the text printer */
    djnCreateConnector(c,0,a,"in",t,"input");

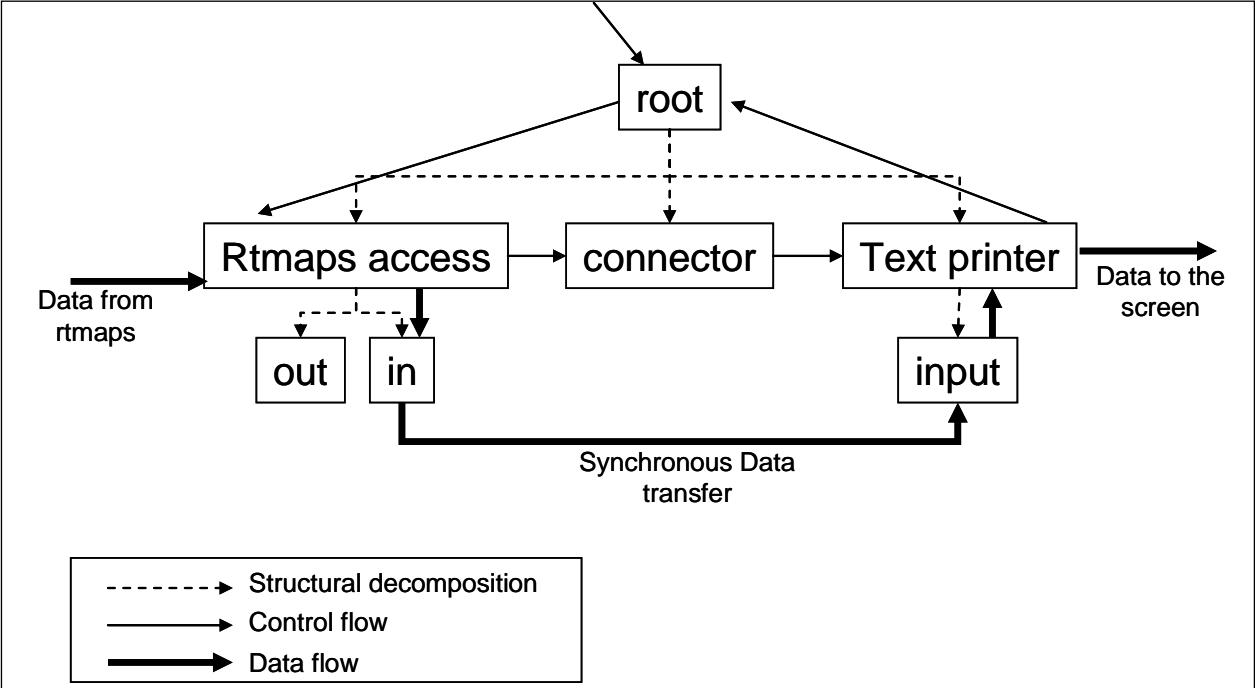
    /* recursive activation of the components */
    djnRunComponent (root);
}
```

**Figure 30 Example of connection with RTMaps**

The structure of the djnn application is shown below.

The application is made of 3 components: an RTMaps access, a connector and a data printer.

The connector enables data transfer: each time a piece of data is received from RTMaps, the connector send it to the text printer.





**Figure 31 Structure of the application**

**Detail of extension 2.2: new mechanisms for verification at runtime**

djnn will also be extended to provide support for dynamic analysis through observers. Observers are finite state-machine components which are used to detect counterexamples of a property. An observer is added to the final application and encodes a set of bad states of an invariant property. During execution time, the observer enters a bad state if and only if the execution so far has broken the property that is being checked.

This approach allows the verification of a certain class of properties: safety properties. Safety properties require that a given event or a given state (typically a “bad” one) never happen. Intuitively, the bad thing is something that can be immediately noticed from an observed behaviour, since the observer enters a bad state after a finite number of steps or not at all, and thus the counterexample for the property must be finite.

The main objective is to be able to define a translation from a property to a djnn construct in order to build the observer.

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

### 2.3.2.2 Application domain and Use Cases

As described above, djnn is currently being improved to prepare it for verification of interactive systems along two axes:

- Extension 1: Static analysis of djnn models: properties are checked on djnn model.
  - Extension 1.1: Internal analysis of djnn models: properties are checked directly on djnn model.
  - Extension 1.2: External analysis: djnn components and djnn platform are translated to an external model on which verification is performed. In HoliDes context, we selected Petri nets as an external target model supported by the GreatSPN tool.
- Extension 2: djnn for dynamic analysis

Up to now, djnn is forecasted to be used in:

WP6: use case 6.7 (Operator task schedule and guidance)

WP7: use case 7.1 (Divergence assistance)

WP9: use case 9.2 (Overtaking)

### 2.3.2.3 Integration in the HF-RTP and interaction with other MTTs

In this release, djnn will be interconnected with:



GreatSPN: model checker of GreatSPN will be used as a complementary way to perform formal verification on djnn model (as described in extension 1.2 above).

For this purpose a model transformation between GreatSPN models and djnn model will be defined and implemented.

Tools integration will be done according to Open Service for Lifecycle Collaboration (OSLC) specification. Although integration with GreatSPN is currently being studied, following major orientations have been decided:

- Translation of djnn components and structure into Petri net model must be done by defining a formal semantic for djnn expressed with state-transition and process model.
- Expression of translated Petri net models into a format that can be understood by GreatSPN. Several normalized formats have been defined



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

for Petri net description, for example PNML, the Petri Net Markup Language

- Exchange of models through OSLC

RTMaps: as described in extension 2, the djnn final application will be interconnected to the RTMaps environment in order to perform dynamic analysis through observers.



Moreover, djnn will be interconnected through OSLC in order to save several data related to verification, for example: results of verification of a requirement, tracing between tools used for verification and the requirements to be verified, date of verification, etc.

### **2.3.3 GreatSPN (University of Torino)**

#### **2.3.3.1 Description and Objectives**

GreatSPN (from University of Torino partner) is a tool with a common graphical interface for a set of formalisms and solvers. The formalisms considered belong to the class of discrete events dynamic systems (DEDS) and include: Petri nets (Place/transition nets with priorities and inhibitor arcs), Coloured Petri nets (the class called Symmetric Nets), their stochastic extensions to include random delays, namely Generalized Stochastic Petri Nets (GSPN) and Stochastic Well-formed nets (SWN), and Markov Decision Petri Nets (MDPN and their extensions to deal with colours – MDWN- and to account for uncertainty - MDPNU).

The analysis for Petri nets and its coloured extensions includes animated simulation, proof for basic properties like boundedness (finite state space) and life of events, state space generation and model-checking of CTL properties. The stochastic extension has an associated set of solvers that differ depending on the distribution of the random delay associated to events: if the random delay is exponential, the set of solvers is the classical set for CTMC - Continuous Time Markov Chain (or for MDP - Markov Decision Processes - in the case of MDPN), to compute the probability of states or of specific conditions in steady-state or at a finite time horizon  $t$ . In case of MDP, GreatSPN is also able to provide the strategy that minimize/maximize a given reward function. Work is undergoing to fully integrate in GreatSPN the solvers of the University of Dortmund to compute strategies for BDMP, the class of MDP with uncertainty that are derived from MDPNU.

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

GreatSPN includes also an innovative stochastic model checker, which allows checking CSLTA (Donatelli, 2009) properties, a superset of the well-known logic CSL (Aziz, 2000). GreatSPN allows also solving DSPN, an extension of GSPN to include, in a restricted manner, events with fixed (deterministic) delays: the resulting process is a Markov Regenerative Process - MRP. When GSPN are extended with delays of general distributions, the tool provides a simulator.

GreatSPN is used in two different flavours in the project.



The first one is the use of MDPN (and MDPNU) and MDP (and BMDP) solutions inside WP9, in particular for the driver model (co-pilot) of the CRF use cases. Currently the use case under examination and development is UC4 (lane change) as described in D9.3. This part requires integration with RTMaps, integration described in Section [2.3.3.3](#).

The second one is the use of GreatSPN as a Verification/Validation engine that will take two different directions: integration in the HF platform so as to work together with other HoliDes tools (as described in Section [2.3.3.3](#)) and direct use of GreatSPN for the verification of requirements in the applicative WPs (as described in Section [2.3.3.2](#)).

### **2.3.3.2 Application domain and Use Cases**

The work in WP2 (MTT\_requirements analysis excel file) has identified 35 requirements in which GreatSPN could be used in the applicative domains of HoliDes. Of the 35, 28 are from automotive and will be accounted for in the specification of the WP9 MDP-based co-pilot, two of them are from WP8 (IREN control room, WP8\_IRN\_CR\_REQ02 and WP8\_IRN\_CR\_REQ20), with notably one requisite that includes also timing constraints, and five of them are instead from WP6 that goes from rather generic, albeit very important ones (like WP6\_PHI\_HEA\_REQ14, that advocates the need for a formal validation of the product) to more specific ones. As for WP8, also in WP6 there is a requirement to provide service within deadlines; moreover there is a special requirement about the validation of the product behaviour in failure situations.

In accordance with the HoliDes global plan, priority was given on the first year to WP9 needs, while we are currently supporting IREN, ReLab and

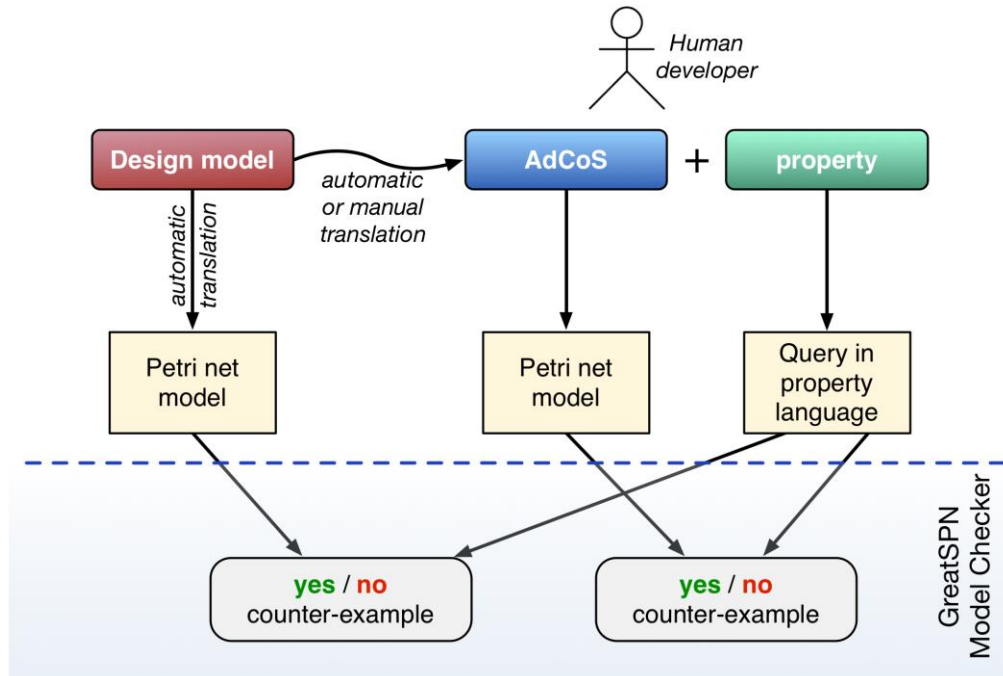
	<p><b>HoliDes</b></p> <p><b>H</b>olistic Human Factors <b>D</b>esign of Adaptive Cooperative Human- Machine Systems</p>	
--	---	---

Airbus in the identification and application of the possible uses of Petri nets for the verification and evaluation of WP8 use cases (utilities control room and control room for border control). In particular IREN and ReLab have identified as initial target of the GreatSPN-based analysis a comparative evaluation of the operator assignment procedure (the current one with respect to the one that will be supported through the new HMI developed in HoliDes).

Figure 32 exemplifies two different ways of working with GreatSPN for the assessment of an application or use case requirement. The figure assumes a property is of a logic type (with yes or no answer) but a similar approach could be applied also for the assessment of a quantitative indices, like the number of calls correctly transferred from the control room to the in-field operators in WP8.

The human developer can develop an AdCoS implementation, or can define a Design Model from which an AdCoS implementation is generated, usually in an automatic or semi-automatic manner (model based life-cycle). A set of properties/requirements is also associated to the AdCoS. The assessment of such properties/requirements is the objective of the Verification and Validation activity. There are two possible approaches. The first one is to automatically derive from the design model a Petri Net model, properties are also translated in Petri net terms and GreatSPN will provide a yes (the property is true) or no (the property is false) answer, or possibly a counterexample. An example of such a verification workflow is the case in which the AdCoS design is realized through UML state charts and sequence diagrams, for which the automatic translation to Petri net is available. Another approach is instead to manually derive a Petri net model from the AdCoS itself, the advantage being that a human may produce more abstract models than an automatic translation (since he may easily include a certain amount of domain knowledge), but with the obvious disadvantage of being a more error prone process since the human is more deeply involved.

At the current stage of work for WP8 application of IREN control room, we expect the Petri net models to be manually created from the AdCoS specification.



**Figure 32: The use of the GreatSPN model-checking facilities**

### 2.3.3.3 Integration in the HF-RTP and interaction with other MTTs

The model checkers of GreatSPN will instead be included in the OSLC-based HF-RTP platform, for integration with djnn, and potentially with other partners' tools.

At the moment only one tool, djnn (see Section 2.3.2), has been surely identified for integration through the HF platform: since the team of djnn is in the process of defining a Petri net semantics for the djnn formalism, it will then be natural to verify djnn properties using the model-checkers of GreatSPN. But we are also investigating the possibility of using GreatSPN to provide a model checker for the task models tools of OFFIS, an objective which is being postponed to the second half of second year.

For the time being the use of GreatSPN in WP8 does not seem to require an interaction (and therefore integration) with other MTTs.

### 2.3.4 Experiment Data Archive (EDA)



EDA is provided by Tecnalía Research & Innovation (TEC) based on the requirements defined by Honeywell (HON). It is an archive tool designed and developed in Python/Django to allow the definition and storage of multiple projects including their full data structure through a web interface and a web services API.

#### 2.3.4.1 Description and Objectives

EDA is a centralized archive for HF data acquired in various development phases of a project. EDA stores acquired data together with meta-information that links the data to the project structure. The projects structure comprehends the phases executed within each project such as requirements, design, implementation or evaluation. Within a phase there are a number of data collection sessions which is the lowest level of meta-information hierarchy.

A data collection session contains a number of data records, obtained through one of the four basic methods. Despite there are numerous variants to the basic methods, from the point of view of data storage, the following classification is applicable:

- *Observation*: activity where the user is observed in his work environment while doing tasks that should be addressed by the new system. The observer makes notes while observing the user and the user can comment on his actions. A camera or a voice recorder can be used to support data analysis.
- *Interview*: a guided process where the subjects are presented a set of open ended questions and they freely elaborate each of them. The questions are forged to direct the subjects to talk about specific aspects of a problem
- *Questionnaire*: allows subjects to select defined options to the presented questions. As in case of interview, the questions are created with an intention to get feedback on specific aspects of a problem. However, the knowledge of the problem is deep enough to state options for answers.

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

- *Experiment*: activity when a subject works out his task in more or less realistic environment. The task is selected to address the problem being studied and usually it is a dynamic process, during which a number of parameters is recorded in time.

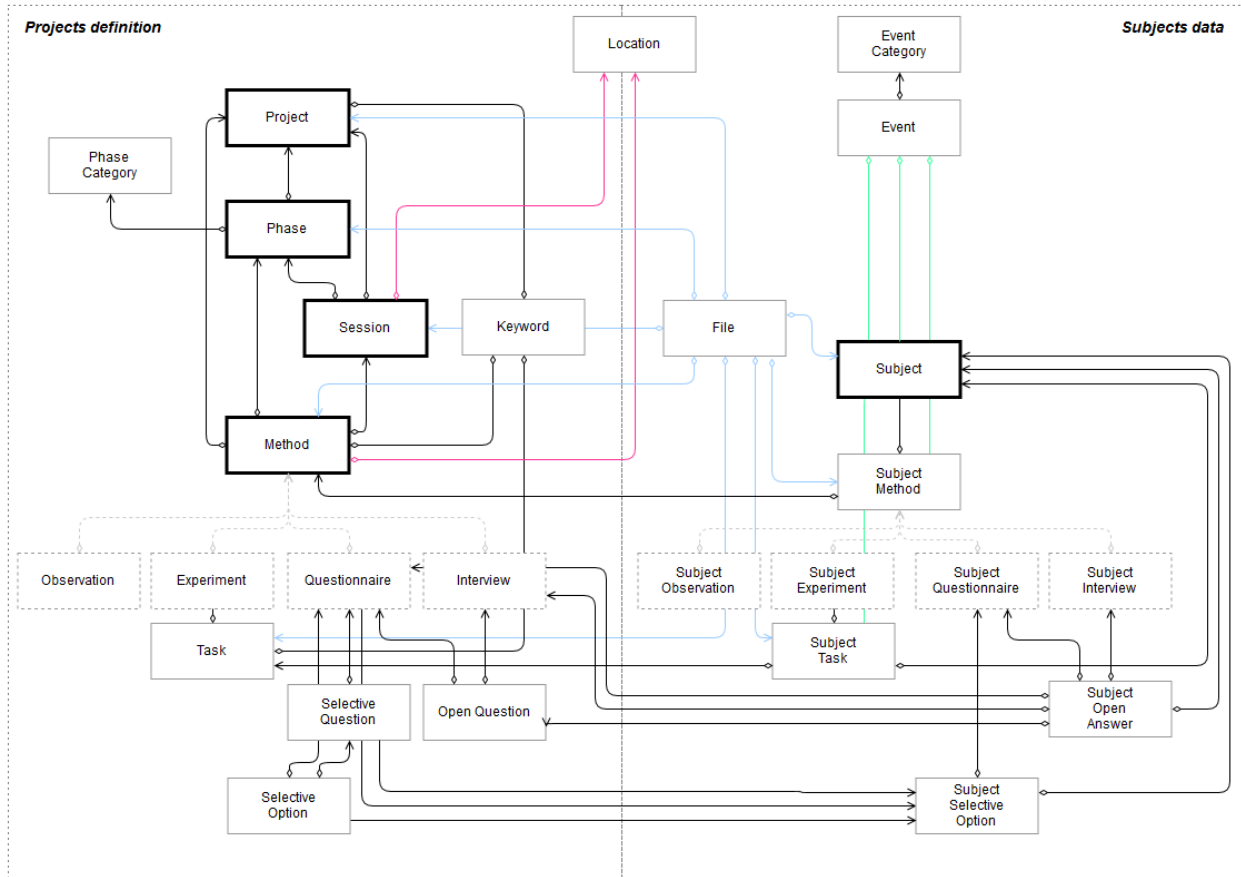
As results, there are recordings of specific parameters. Many of them create a time series of parameter's value (such as ECG record or simulator log). Additionally video and voice can be recorded and notes taken by the experimenter can be compiled as a list of events (annotations). The recordings are linked to subjects and then placed in the context of a project following the data model of EDA, see Figure 33.

The tool defines a set of public methods to be consumed as web services along with an FTP to allow integration with other tools. It has been integrated with RTMaps to allow the archiving of the files generated during an experiment within a project.



# HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems





**Figure 33: EDA data model**

Data is always collected within a frame of defined activities – in an experiment session during a defined phase of a project. When stored to database, each record is described with meta-information that defines its position in such framework.

At the low level of database structure, the meta-information considers acquisition time, keywords, author, subjects. At higher level the records are organized in data collections within a project. Going from the top, the data hierarchy starts at definition of project.

The *project* contains keywords describing what the project is about. The project has start and end date and a project team, which is for purposes of the database represented by team leader, HF focal and SW lead. When using the database, these three persons can provide full details on the project.

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

Person from a project team may be representing just by a contact, no further information is needed.

Each project executes a number of *phases* – such as requirements, design, implementation, evaluation. Data are always collected within one phase and each phase has specific requirements on what purpose the data collection has, what methods are suitable and what data should be collected. The project phase also defines artifacts that should be created in the phase. These artifacts should be attached to the project phase as file-type record.

Within a phase there is a number of data collection sessions – there may be more different experiments or an experiment needs to be repeated. A data collection session is the lowest level of meta-information hierarchy. The data collection session contains keywords that specify what is being investigated, what methods are used etc. Each *session* has a start and end date and contains two files – data collection plan describing what and how it will be done and report/summary file describing results of analysis of the acquired data.



A data collection contains a number of data records – being observations, interviews, questionnaires or experiments.

#### **2.3.4.2 Application domain and Use Cases**

This EDA tool is been designed for the aeronautic application domain (related to WP7) to keep all the relevant information about experiments set previously in RTMaps. The particular Use Case where EDA tool is being used is the Diversion Assistant (DivA AdCoS).

During the course of a project, there are a number of activities where data from subjects are collected. Based on the phase of the project, data is used for definition of a concept (voice of a customer) and requirements. Later verification process collects evidence that the developed system meets the requirements and yet later the system undergoes a series of evaluations to prove the system being acceptable to the users and useful for executing their tasks. In the end, the system is validated to meet regulations and architectural constrictions.



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



Along with these large size data collection events, the system is usually being tested throughout the whole development process and the assembled data is analysed to improve the system.

The tool is designed to store projects structure and the data generated when confronting these projects with subjects in a centralized archive.

#### **2.3.4.3 Integration in the HF-RTP and interaction with other MTTs**

Currently there is a connection with RTMaps to store the video, audio and/or CSV files generated during an RTMaps experiment in EDA.

It is now to be decided whether EDA should provide OSLC integration capabilities to expose metadata and files from the projects with other tools via OSLC service providers.

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

**The next sections of the Deliverable are  
confidential.**

**They are only available in the  
"Confidential Version" of D4.6.**



## HoliDes

Holistic Human Factors Design of  
Adaptive Cooperative Human-  
Machine Systems

## References

- Aziz, A. S. (2000). Model-checking continuous-time Markov chains. . *ACM Trans. Comput. Log.* 1(1), 162–170.
- Bause, F et al (1995) Abstract Petri Net Notation, Petri Net Newsletter, No 49 (1995) 9-27.
- Beccuti M., Franceschinis G. (2007). Markov Decision Petri Net and Markov Decision Well-formed Net formalisms. *28th Int. Conference on Applications and Theory of Petri Nets and other Models of Concurrency 2007. 4546*, pp. 43-62. LNCS.
- Beccuti M. et al (2011). MDWNSolver: a framework to design and solve Markov Decision Petri Nets. (R. Consultants, Éd.) *International Journal of Performability Engineering*, 417-428.
- Bellet, T. et al. (2009). A theoretical and methodological framework for studying and modelling drivers' mental representations. *Safety Science*, 47, pp. 1205–1221.
- Bellet, T. M. (2012). A computational model of the car driver interfaced with a simulation platform for future Virtual Human Centred Design applications: COSMO-SIVIC. *Engineering Applications of Artificial Intelligence*, 25, pp. 1488-1504.
- Chatty, S. and Magnaudet, M. and Prun, D. (2015). Verification of properties of interactive components from their executable code. To appear in *The 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2015)*.
- Donatelli et al. (2009). Model checking timed and stochastic properties with CSLTA. *IEEE Trans. Software Eng.* 35(2), 224–240.
- Dufourd C. et al. (1998). Reset nets between decidability and undecidability. *25th International Colloquium on Automata, Languages and Programming (ICALP'98)*, 1443, 103-115.
- Gruyer D. et al (2010). SiVIC, a virtual platform for ADAS and PADAS prototyping, test and evaluation. *FISITA'10*. Budapest.
- Gruyer D., et al (2011). Distributed Simulation Architecture for the Design of Cooperative ADAS. *FAST-ZERO (Future Active Safety Technology)*. Tokyo.
- Jensen, K. (1996). *Coloured Petri Nets*. Berlin: Heidelberg.



## HoliDes

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

HoliDes

- Kwiatkowska M. et al. (2011). PRISM 4.0: Verification of Probabilistic Real-time Systems. *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)* (pp. 585-591). Springer.
- OMG (2015) Data Distribution Service (DDS), Version 1.4, April 2015, - <http://www.omg.org/spec/DDS/1.4>
- Prun, D. and Magnaudet, M. and Chatty S. (2015) Towards Support for Verification of Adaptive Systems with djnn. *The Seventh International Conference on Advanced Cognitive Technologies and Applications. IARIA*, ISBN: 978-1-61208-390-2, pp. 191-194
- Puterman, M. (2005). *Markov Decision Processes*. Chichester: Wiley.
- Torino, University of (2015.). [www.di.unito.it/~greatspn](http://www.di.unito.it/~greatspn).
- Puterman, M. (2005). *Markov Decision Processes*. Chichester: Wiley.
- W3C. (2010) XML path language (XPath) 2.0 (second edition). <http://www.w3.org/TR/xpath20/>.
- Weber, M. and Kindler E. (2003) The Petri Net Markup Language, Petri Net Technology for Communication Based Systems, LNCS Vol 2472, Springer-Verlag, Berlin Heidelberg Newyork 2003