


	<p>HoliDes</p> <p>Holistic Human Factors Design of Adaptive Cooperative Human- Machine Systems</p>	
<p>D4.4 - Techniques and Tools for Model-based Analysis Vs1.0 incl. Handbooks and Requirements Analysis Update</p>		

Project Number:	332933
Classification:	Public Version
Work Package(s):	WP 4
Milestone:	M2
Document Version:	V1.0
Issue Date:	<31.12.2014>
Document Timescale:	Project Start Date: October 1, 2013
Start of the Document:	M14
Final version due:	M15
Deliverable Overview:	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.
Keywords:	Model-based Analysis, Simulation Techniques and Tools, AdCoS Evaluation, Tailored HF-RTP
Compiled by:	T. Bellet, JC Bornard (IFS), D. Prun (ENA)
Authors:	T. Bellet (IFS), D. Prun (ENA), N. Dulac (INT) D. Gruyer, J.C. Bornard, B. Richard (IFS), A. Smrcka (BUT), S. Donatelli, E. Amparore, M. Beccuti (UTO), S. Rieger (TWT), J.P. Osterloh (OFF)
Reviewers:	R. Leblond (Airbus) & F. Tango (CRF)



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

Technical Approval:	Jens Gärtner, Airbus Group Innovations
Issue Authorisation:	Sebastian Feuerstack, OFFIS

© All rights reserved by HoliDes consortium

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

RECORD OF REVISION		
Date	Status Description	Authors
15.12.2014	Initial version (Version.01)	T. Bellet (IFS), D. Prun (ENA), N. Dulac (INT)
13.01.2015	ENA contribution	D. Prun (ENA)
14.01.2015	BUT contribution	A. Smrcka (BUT)
15.01.2015	IFS contribution	T. Bellet, D. Gruyer, JC Bornard, B. Richard (IFS)
16.01.2015	UTO contribution	S. Donatelli, E. Amparore, M. Beccuti (UTO)
19.01.2015	Integrated Version.02	T. Bellet (IFS)
21.01.2015	TWT contribution regarding OSLC integration (Version.03)	S. Rieger (TWT)
21.01.2015	OFF contribution (version.04)	JP Osterloh (OFF)
22.01.2015	UTO MTT update (version.05)	S. Donatelli, E. Amparore, M. Beccuti (UTO)
23.01.2015	BUT MTT update	A. Smrcka (BUT)
28.01.2015	Integrated Version.06	T. Bellet, JC Bornard (IFS)
30.01.2015	Final Version.07 for reviewers	T. Bellet, JC Bornard (IFS)
09.02.15	Reviewed version 1	R. Leblond (Airbus)

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

11.02.2015	Reviewed version 2	F. Tango (CRF)
12.02.2015	Final version	T. Bellet, JC Bornard (IFS)



Table of Contents

- 1 Introduction: WP4 objectives and MTT 9**
- 2 Handbook of Techniques and Tools for Model-based Analysis..... 13**
 - 2.1 Human Operator Models13
 - 2.1.1 COSMODRIVE and COSMO-SIVIC (IFS)13
 - 2.1.1.1 Model Description and Objectives.....13
 - 2.1.1.2 Application domain and Use Cases14
 - 2.1.1.3 Integration in the HF-RTP and interaction with other MTT...15
 - 2.1.2 CASCaS (OFF)18
 - 2.1.2.1 Description and Objectives.....18
 - 2.1.2.2 Application domain and Use Cases19
 - 2.1.2.3 Integration in the HF-RTP and interaction with other MTT...19
 - 2.1.3 MDP-based Co-pilot (UTO)20
 - 2.1.3.1 Description and Objectives.....20
 - 2.1.3.2 Application domain and Use Cases21
 - 2.1.3.3 Integration with HF-RTP and interaction with other MTT21
 - 2.2 MTT for AdCoS modelling and simulation25
 - 2.2.1 RTMaps (INT)25
 - 2.2.1.1 Description and Objectives.....25
 - 2.2.1.2 Application domain and Use Cases26
 - 2.2.1.3 Integration in the HF-RTP and interaction with other MTT...26
 - 2.2.2 Pro-SIVIC (CVT)27
 - 2.2.2.1 Description and Objectives.....27
 - 2.2.2.2 Application domain and Use Cases28
 - 2.2.2.3 Integration in the HF-RTP and interaction with other MTT...29
 - 2.3 MTT for AdCoS design, verification and validation30
 - 2.3.1 AnaConDa, Race Detector, Healer, SearchBestie (BUT)30
 - 2.3.1.1 Description and Objectives.....30
 - 2.3.1.2 Application domain and Use Cases34
 - 2.3.1.3 Integration in the HF-RTP and interaction with other MTT...34



- 2.3.2 Djnn (ENA)35
 - 2.3.2.1 Description and Objectives.....35
 - 2.3.2.2 Application domain and Use Cases38
 - 2.3.2.3 Integration in the HF-RTP and interaction with other MTT ...38
- 2.3.3 GreatSPN (UTO)39
 - 2.3.3.1 Description and Objectives.....39
 - 2.3.3.2 Application domain and Use Cases40
 - 2.3.3.3 Integration in the HF-RTP and interaction with other MTT...41
- 3 Integration Plan: HF-RTP tailoring in WP4.....43**
 - 3.1 WP4 Strategy for integration plan43
 - 3.2 HF-RTP based on OSLC.....43
 - 3.2.1 A brief overview of OSLC43
 - 3.2.2 HF-RTP based on OSLC: Perspective of use for AdCoS43
 - 3.2.2.1 Requirements Tracing43
 - 3.2.2.2 Configuration Management43
 - 3.2.2.3 Reporting43
 - 3.2.3 Tool integration outside the scope of OSLC43
 - 3.3 HF-RTP based on RTMaps.....43
 - 3.3.1 RTMaps Tool chain for AdCoS design, development, simulation and evaluation.....43
 - 3.3.1.1 Connected tools43
 - 3.3.1.2 AdCoS development toolchain based on RTMaps43
 - 3.3.1.3 RTMaps for AdCoS verification and validation: IDEEP functions 43
 - 3.3.2 Use of HF-RTP based on RTMaps for AdCoS design and evaluation: application to automotive domain.....43
 - 3.3.2.1 HF-RTP based on RTMaps for ADAS and AdCoS modelling ..43
 - 3.3.2.2 Main features to support AdCoS dynamic simulation43
 - 3.3.2.3 Toward a first tailored HF-RTP based on RTMaps for AdCoS verification and validation for automotive domain43
 - 3.3.2.4 Illustration of RTMaps application previously used for real and virtual ADAS/ADCOS design and test43



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



- 4 WP4 MTT requirements updating44**
 - 4.1 Requirements towards HF-RTP44
 - 4.1.1 COSMODRIVE and COSMO-SIVIC44
 - 4.1.2 CASCaS.....44
 - 4.1.3 MDP based Co-pilot44
 - 4.1.4 GreatSPN.....44
 - 4.1.5 Djnn44
 - 4.1.6 AnaConDa, Race Detector, Healer and SearchBestie44
 - 4.2 Requirements towards AdCoS44
 - 4.2.1 COSMODRIVE and COSMO-SIVIC44
 - 4.2.2 CASCaS.....44
 - 4.2.3 MDP based Co-pilot44
 - 4.2.4 GreatSPN.....44
 - 4.2.5 Djnn44
 - 4.2.6 AnaConDa, Race Detector, Healer and SearchBestie44
 - 4.3 Requirements towards other MTT44
 - 4.3.1 COSMODRIVE and COSMO-SIVIC44
 - 4.3.2 CASCaS.....44
 - 4.3.3 MDP based Co-pilot44
 - 4.3.4 GreatSPN.....44
 - 4.3.5 Djnn44
 - 4.3.6 AnaConDa, Race Detector, Healer and SearchBestie44
- 5 Conclusion and perspectives44**
- 6 References45**



HoliDes

Holistic Human Factors Design of
Adaptive Cooperative Human-
Machine Systems

holiDES

List of figures

Figure 1: WP4 MTT for AdCoS Design, Verification and Validation	11
Figure 2: Overview of COSMODRIVE Model	13
Figure 3: IFS-SiVIC platform for ADAS simulation	15
Figure 4: COSMO-SIVIC V-HCD use for AdCoS evaluation	16
Figure 5: V-Design process of AdCoS with the COSMO-SIVIC platform	17
Figure 6: Structure of the cognitive architecture CASCaS	18
Figure 7 The GreatSPN GUI while playing the token game	22
Figure 8: integration of MDP in RTMaps	24
Figure 9 : The RTMaps Studio	25
Figure 10: The ProSIVIC simulator	27
Figure 11: ProSIVIC interoperability with RTMaps	29
Figure 12: A simple analyser monitoring lock operations	32
Figure 13: An example of different noise settings for reads and writes.	32
Figure 14: A scheme of usage RaceDetector & Healer	33
Figure 15: Examples of bindings definitions in djnn	35
Figure 16: examples of derived control structure definitions in djnn	36
Figure 17: Example of a FSM definition	36
Figure 18: example of connection with FSM	37
Figure 19: Intermediate integration scenario where WP4 tools are integrated by domain-specific generic OSLC adaptors	46
Figure 20: Integration scenario where WP4 tools are integrated directly with tool-specific OSLC adaptors	47
Figure 21: RTMaps tools connexions	49
Figure 22: Matlab Simulink	50
Figure 23: QtCreator, a graphical editor for Qt and QML GUIs	50
Figure 24: An AdCoS development tool chain	51
Figure 25: I-DEEP functions of RTMaps	53
Figure 26: ADAS and AdCoS Modelling with RTMaps	54
Figure 27: Overview of the tailored HF-RTP based on RTMaps	56
Figure 28: QML graphical user interface	57
Figure 29: Stereovision algorithm in RTMaps	57
Figure 30 : Synchronized real-time playback application	58
Figure 31: The use of the GreatSPN model-checking facilities	63





HoliDes

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



Executive Summary

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. 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 MTT currently identified to be used in WP4. Section 3 provides a description of the integration plan and a first example of tailored HF-RTP(s) interconnecting WP4 MTT, in order to support AdCoS validation and verification. Section 4 is dedicated to MTT requirements updating, according to the progress done during the first phase of the project. Lastly, conclusion will introduce the next steps of WP4 for the next phases of the project.

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



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**¹ (answering to 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 to 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 uses case, for instance) and/or by considering individually some specific

¹ “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).

	<p>HoliDes</p> <p>Holistic Human Factors Design 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, 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 stimulations 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 Model-based Technics and Tools (MTT) 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 Human Factor Models-based Technics and simulation Tools (pre-existing or specifically designed in WP2 and WP3), interconnected in a tailored HF-RTP (to be 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 to 9).



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

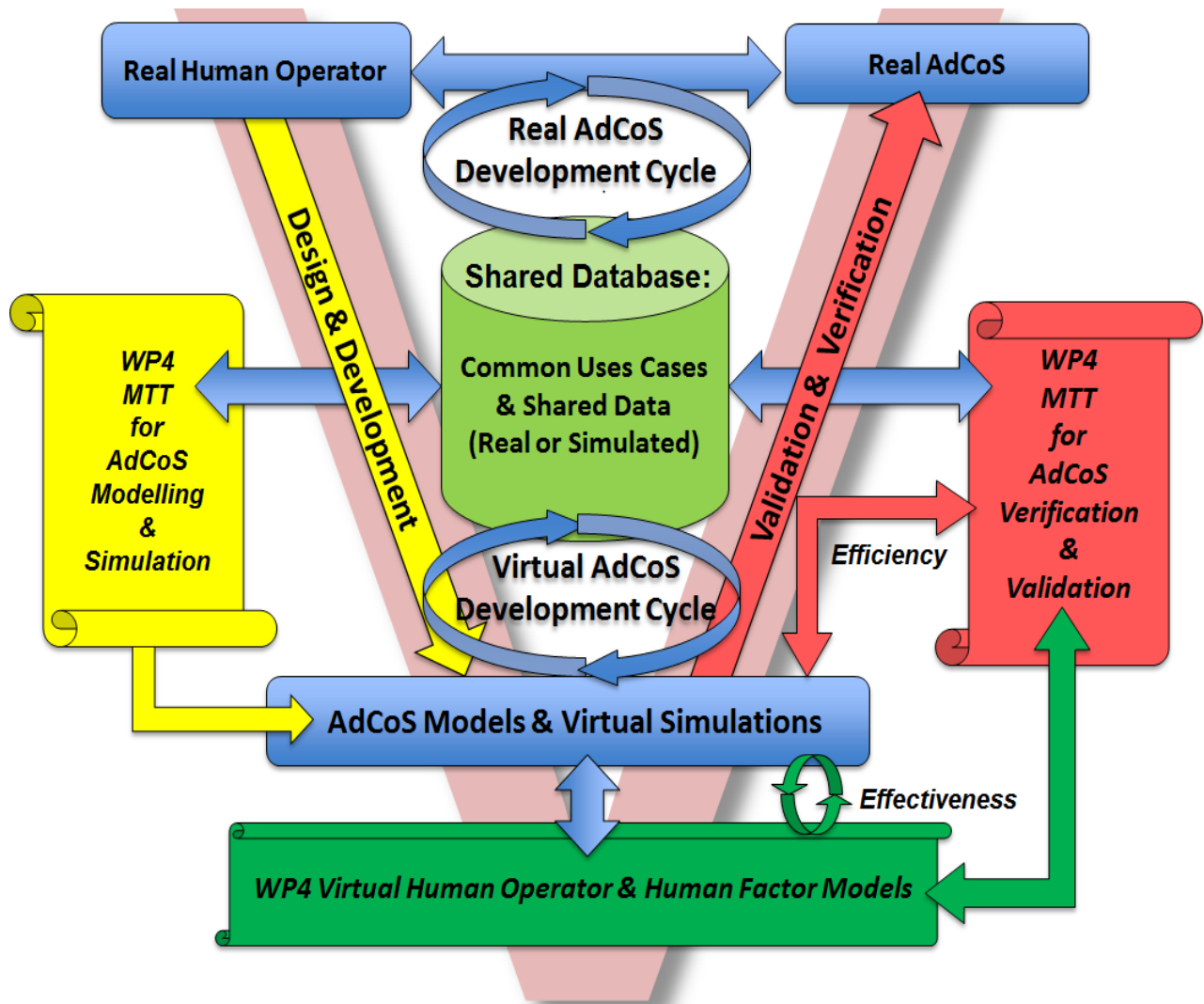



Figure 1: WP4 MTT for AdCoS Design, Verification and Validation

During the first year of the project, a first set of MTT have been already identified for jointly supporting this future AdCoS design and evaluation

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

process. However, other MTT, currently under development in the various HoliDes WPs will be also liable to be progressively integrated later this chain, according to future needs potentially dependant of applications domains.

Currently, as presented in Figure 1, three main types of MTT 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 MTT 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 3 groups of complementary MTT to be used in WP4 are successively presented in the next section of this deliverable.



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 (IFS, OFF, UTO), have been currently identified to support WP4 objectives: COSMODRIVE, CASCaS and the MDP-based Copilot.

2.1.1 COSMODRIVE and COSMO-SIVIC (IFS)

2.1.1.1 Model Description and Objectives

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

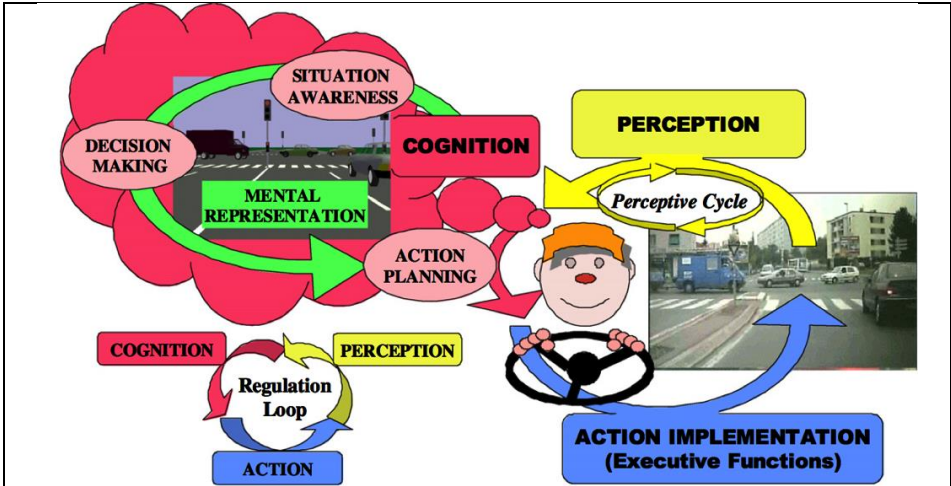


Figure 2: Overview of COSMODRIVE Model

Through this main regulation loop, the model allows to:

- Simulate human drivers perceptive functions, in order to visually explore the road environment (i.e. *perceptive cycle* based on specific driving knowledge called "schemas"; (Bellet T. B.-A., 2009) (Bellet T. M., 2012)) and then to process and integrate the collected visual pieces of information in the Cognition Module.



HoliDes

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

hOLIDES

- Simulate 2 core cognitive functions that are (i) the elaboration of *mental representations* of the driving situation (corresponding to the driver's Situational Awareness; (Bellet T. B.-A., 2009)) and (ii) a *decision-making* processes (based on these mental models of the driving situation, and on an *anticipation process* supported by dynamic mental simulations)
- Implement the driving behaviours decided and planned at the cognitive level, through a set of effective actions on vehicle commands (like pedals or steering wheels), in order to dynamically progress along a driving path into the road environment.

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

2.1.1.2 Application domain and Use Cases

COSMODRIVE is dedicated to AdCoS design in the automotive domain. In the frame of WP2, a new version of this model (described in D2.4) was specifically developed for HoliDes project objectives, in order to be used in WP4 (and then in WP9 as a simulation demonstrator) for the Virtual Human Centred Design (V-HCD) of future AdCoS. In the "HF-RTP" logic of HoliDes, COSMODRIVE will play the role of one of the "Human Factor" (HF) models (focused on car driving), interacting with a virtual AdCoS to be virtually simulated by the HF-RTP.

In the frame of WP4, the aim is to use the HF-RTP approach – as a Virtual Human Centred Design process – for the virtual design and validation of a specific AdCoS currently developed by IFSTTAR in WP3. This AdCoS will be supported by *MOVIDA* functions (Monitoring of Visual Distraction and risks Assessment). Synthetically, it is an integrative co-piloting system supervising several simulated Advanced Driving Aid Systems (ADAS) to be managed in and Adaptive and Cooperative way by *MOVIDA* algorithms, according to the drivers' visual distraction states and to the situational risks assessment (detailed description available in D3.4).



To support this global objective, COSMODRIVE have been interfaced in WP4 with the Pro-SiVIC platform, as used at IFSTTAR-LIVIC (Gruyer D. G. S., 2010) (Gruyer D. G. S., 2011) to simulate Advanced Driving Aids Systems (Figure 3), and will be completed with MOVIDA functions in HoliDes.

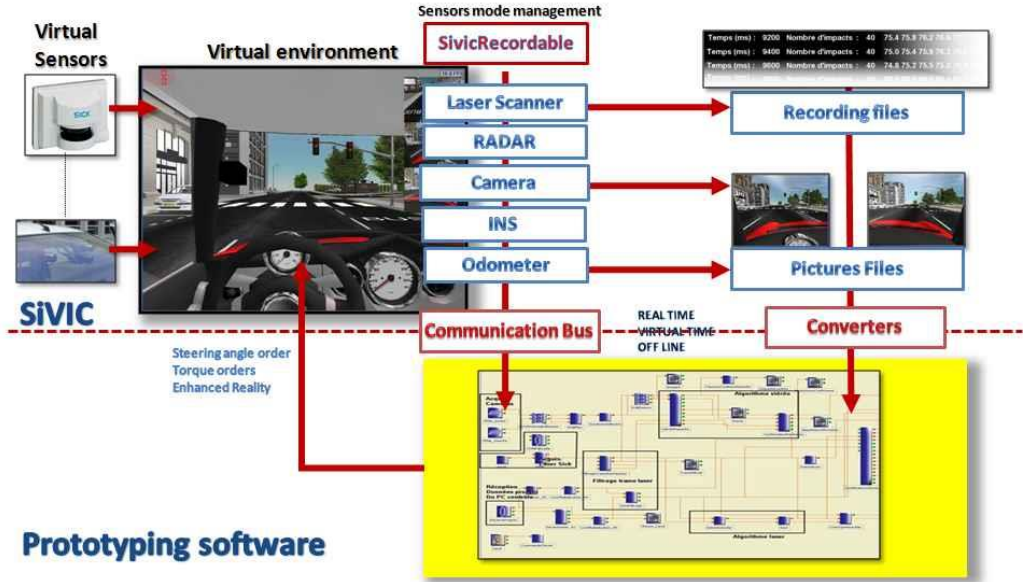


Figure 3: IFS-SiVIC platform for ADAS simulation

2.1.1.3 Integration in the HF-RTP and interaction with other MTT

In addition, COSMO-SIVIC has been also connected in WP4 with RTMaps and Pro-SiVIC software, in order to have a Virtual Human Centred Design platform for AdCoS design and evaluation. Figure 4: provides an overview of this future V-HCD platform to be progressively designed and developed by IFS during the project, in partnership with CIVITEC and INTEMPORA.

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 MOVIDIA functions (simulated through SIVIC, Pro-SIVIC and RTMaps) into (4) a virtual 3D road environment (simulated with Pro-SIVIC).



HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

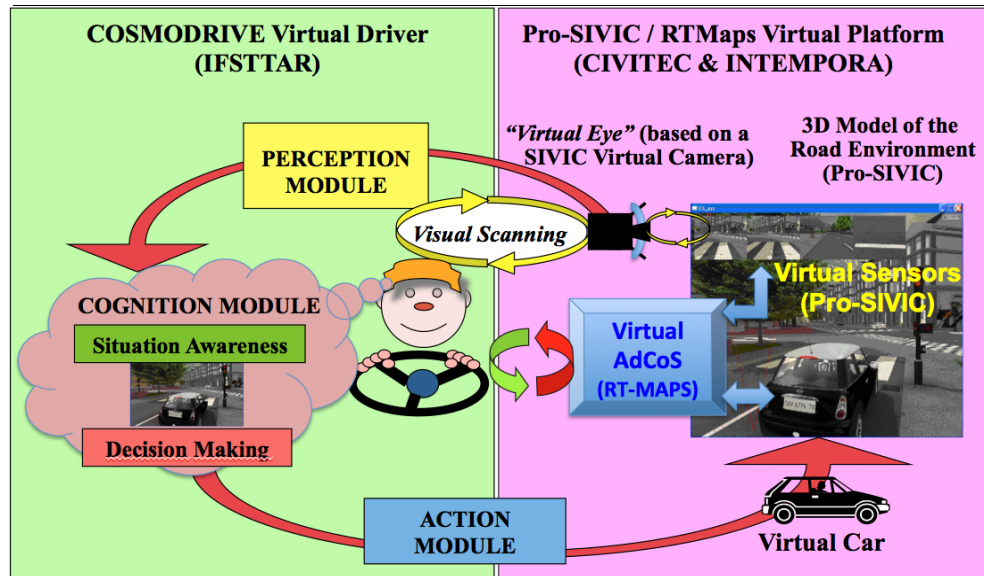


Figure 4: COSMO-SIVIC V-HCD use for AdCoS evaluation based on dynamic simulation (supported by Pro-SIVIC and RTMaps software)

In WP4, the aim is to use this COSMO-SIVIC Virtual Human Centred Design platform (V-HCD) to support AdCoS validation and verification from dynamic simulation, by considering the future use of the MOVIDA-AdCoS by human drivers (i.e. end-users, as simulated with COSMODRIVE).

The following figure presents the long term perspectives of COSMO-SIVIC use for future AdCoS virtual design and evaluation, supported on the HF-RTP approach. In this "V design process", only the steps integrated under the red line will be effectively implemented by IFS during HoliDes project, with the aim to validate and demonstrate in WP9 the advantage of using a tailored HF-RTP to support AdCoS virtual design in automotive domain (i.e. no real AdCoS for real car will be developed by IFS).

However, the expected advantages in using this Human Factor Model-based design process during HoliDes are to integrate end-users needs since 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 a human drivers *With* and *Without* driving aids (from normal behaviours to critical behaviours due to visual distraction), in order to support the AdCoS and MOVIDA functions virtual design and evaluation at 2 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 accident. Through these simulations, it will be possible to provide ergonomics specifications of human driver needs, as a set of "Critical Scenarios" and "Use Cases" of reference, liable to be stored in a "reference database".

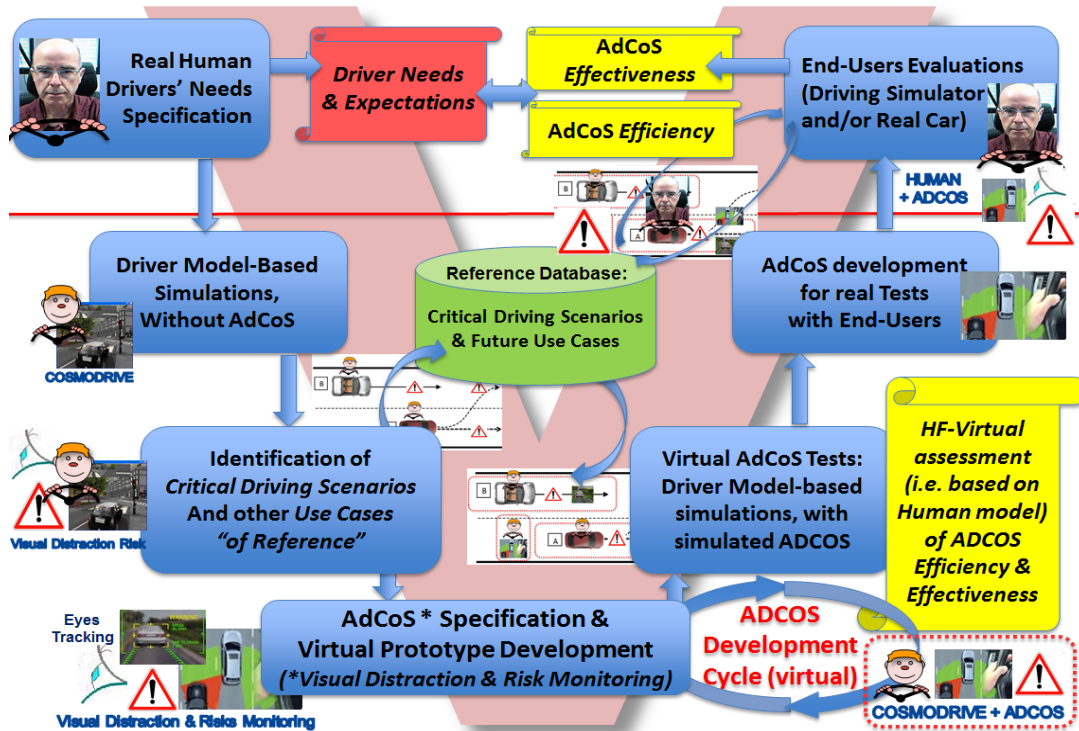


Figure 5: V-Design process of AdCoS with the COSMO-SIVIC platform

Then, during the virtual evaluation process of the MOVIDA-AdCoS, this reference database associated with visual distraction simulations based on COSMODRIVE, will be used to progressively increase its *efficiency* (i.e. AdCoS developments and virtual tests Cycle on the figure) in accordance with the different critical scenarios previously identified. Such HF Model-based simulations will also allow us to assess the potential *effectiveness* of MOVIDA, before effectively developing a real prototype for testing it among real human drivers, through full scale evaluation with end-users on driving simulators and/or with real cars.

2.1.2 CASCaS (OFF)

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.

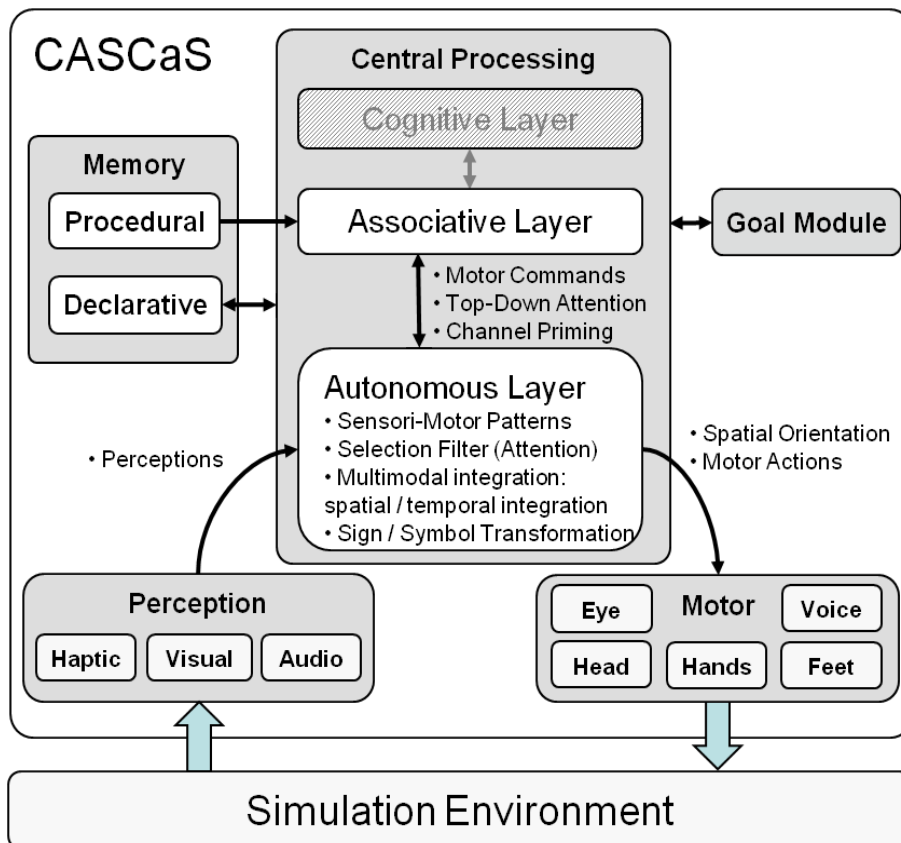




Figure 6: Structure of the cognitive architecture CASCaS with all internal components and the major data flows.

Figure 6 shows the current version of the architecture with all its components. Basically, the architecture consists of 5 components: a *Goal Module* which stores the intentions of the model (what it wants to do next). The *Central Processing* is subdivided into three different layers: the cognitive layer which can be used to model problem solving, the associative layer

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

executes learned action plans and the autonomous layer simulates highly learned behaviour. The memory component is subdivided into a procedural (action plans) and a declarative knowledge (facts) part. The *Perception* component contains models about physiological characteristics of the visual, acoustic and haptic sensory organs, for example models about peripheral and foveal vision. To interact with the external environment the *Motor* component of CASCaS contains models for arm, hand and finger movements. It also comprises a calculation for combined eye / head movements that are needed to move the visual perception to a specific location. In general the model starts observing its environment via the perception and receives input which is stored in the memory component. Depending on its current intention and on the perceived information from the environment, it selects action plans and tries to achieve its current goal. It may generate new goals and further actions, which can be triggered by events perceived from the environment or the model may itself create new goals, based on its own decision making process, to initiate a certain behaviour.

2.1.2.2 Application domain and Use Cases

CASCaS will be applied in WP9 in cooperation with TAK for evaluation of their User Interface/AdCoS. For this, CASCaS will be enhanced to calculate Saliency Maps for the user interface. Details of this are still under discussion, and the concrete work on this is planned for the second year.

2.1.2.3 Integration in the HF-RTP and interaction with other MTT

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 is not appropriate. Anyway, the surrounding tools for producing the needed input for CASCaS (e.g. MagicPED), controlling the simulation (e.g. CoSimECS) or processing the output (e.g. Excel, Knime, R) are candidates for integration with the HF-RTP. HF-RTP integration of MagicPED, which is described in D2.4, has been started. Development of CoSimECS is currently postponed to end of second year of HoliDes.

2.1.3 MDP-based Co-pilot (UTO)

2.1.3.1 Description and Objectives

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

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

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

The main features of MDPNs are the possibility to specify the general behaviour as a composition of the behaviour of several components (some of which are subject to local non deterministic choice, and are thus called controllable, while the others are called non controllable); moreover any non-deterministic or probabilistic transition of an MDP can be composed by a set of non-deterministic or probabilistic steps, each one involving a subset of

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

components. Hence, an 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 the non-deterministic (ND) and probabilistic (PR) behaviour respectively).

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.

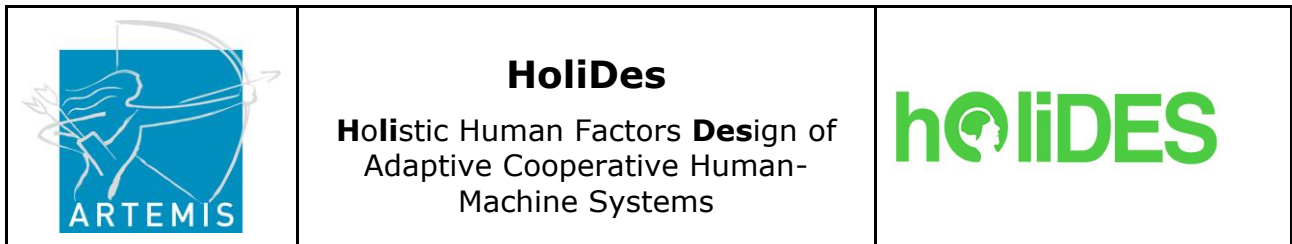
Since the MDPN formalism does not provide MDP with uncertain behaviour, we extend the definition as follows: The uncertainty is introduced at the level of the transition rates in PN^{pr} . In this way, the MDPN underlying process becomes an MDP with incomplete or uncertain transition rates. This extension has been developed for Holidés' needs and is described in deliverable D2.4.

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 involves the CRF test vehicle. 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.

2.1.3.3 Integration with HF-RTP and interaction with other MTT

The co-pilot core is an MDP solver, which is part of the GreatSPN tool. The solver will be part of the run-time environment of RTMaps. Both GreatSPN and RTMaps will be used in the activity of model-based verification of the co-pilot, typical of WP4.



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. Properties can be verified for (colored) Petri net models, but GreatSPN does not yet include a model checker for MDPN. Nevertheless GreatSPN allows to export MDPN (or better, the MDP generated from the MDPN) to PRISM (Kwiatkowska M., 2011), a well-known tool for probabilistic verification.

The integration with RTMaps. The use of RTMaps for the validation of the co-pilot is strictly related to the fact that the co-pilot, at run-time, acts on the data provided by the RTMaps platform, and therefore also in tuning/testing and verification phase the integration with RTMaps is a central node for the co-pilot verification. We revise therefore first the integration for the run-time co-pilot, which is part of the work in WP2 and WP9, and then discuss plan for the verification activity, which is more specific to WP4.

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

Figure 8 shows the structure of the RTMaps component that contains the MDP solver of GreatSPN.

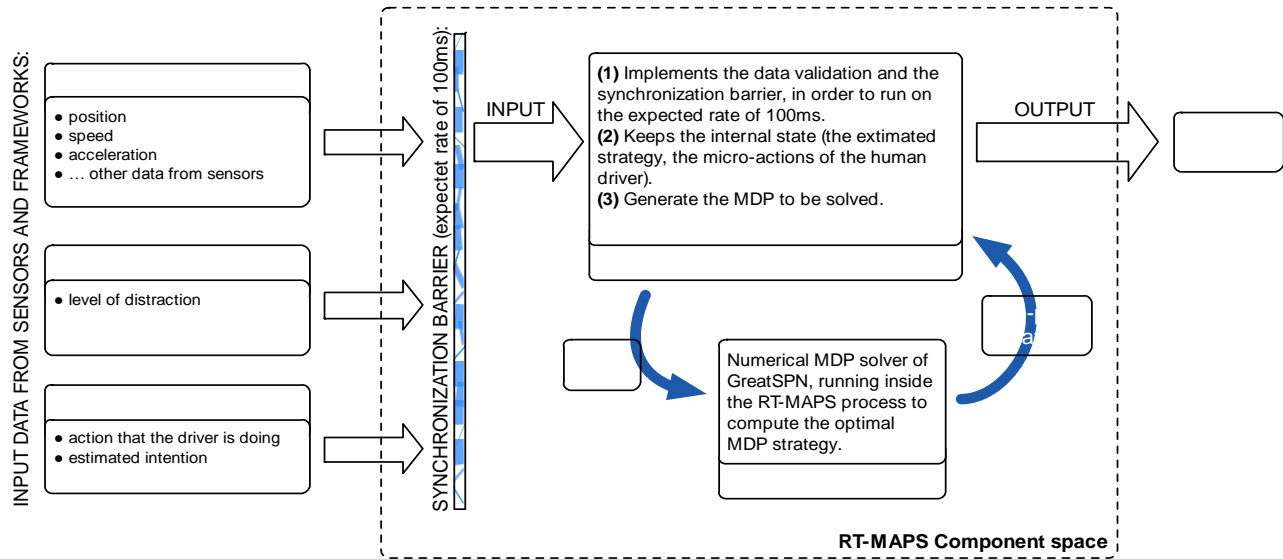


Figure 8: integration of MDP in RTMaps

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

Validation of a software component that is strongly dependent on the input values might not be simply achieved by classical model-checking activities as advocated at the beginning of this section. We plan therefore a validation activity of the MDPN model “plugged” into its run-time environment. The current idea is to use the ability of RTMaps to playback sensor traces in conjunction with the ability of the GreatSPN interface to interact with the MDP solver (token-game, as described above). This approach is similar to the use of RTMaps for the design of WP9 components, as described in section 3.3.2.



2.2 MTT for AdCoS modelling and simulation

Two main Tools, provided by 2 HoliDes partners (INT, CVT), have been currently identified to support WP4 objectives: RTMaps and ProSIVIC.

2.2.1 RTMaps (INT)

2.2.1.1 Description and Objectives

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.) and is connected with many different software (like Matlab/Simulink & dSPACE, Qt / QML, or Pro-SiVIC)

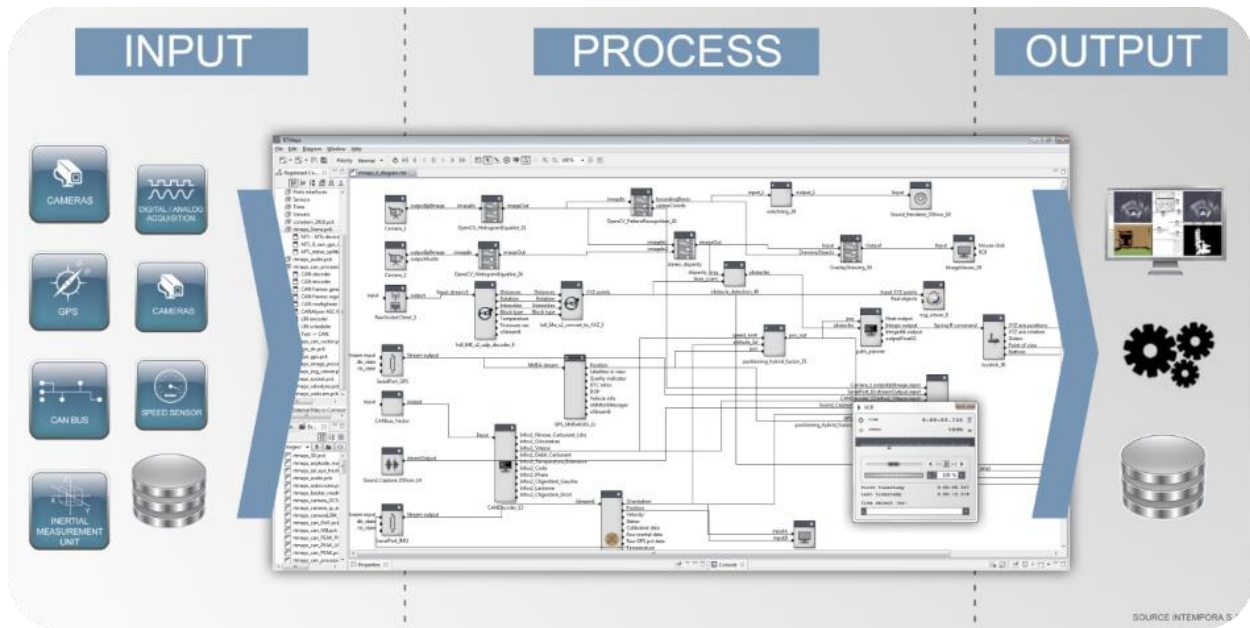


Figure 9 : The RTMaps Studio

RTMaps provides accurate time stamping for each and every data sample entering the application and operates as a multi-threaded environment to be able to manage different data streams with different frame rates, including



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

RTMaps provides also a C++ SDK, with example and tutorial, allowing internal development in order to connect other tools through different communications protocol.

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.

Even if this tool was mainly used in the past for automotive application, it is also usable for other application domains in HoliDes.

2.2.1.3 Integration in the HF-RTP and interaction with other MTT

RTMaps is already used in WP4 as an integrative tool to support a tailored HF-RTP including:

- Connexion to Pro-SiVIC for environment and sensor simulation
- Connexion to COSMODRIVE and COSMO-SIVIC, build with RTMaps SDK, for human driver simulation and virtual ADAS modelling.
- IFS AdCoS based on MOVIDA (MOnitoring of VIsual Distraction and risks Assessment) system is also connected
- Connexion to GreatSPN and MDP-based Copilot, to design AdCoS demonstrator for real car (CRF)

A detailed description of a first tailored HF-RTP based on RTMaps for AdCoS design, development, verification and validation is described in section 3.3.



HoliDes

Holistic Human Factors Design of
Adaptive Cooperative Human-
Machine Systems

holiDES

2.2.2 Pro-SIVIC (CVT)

2.2.2.1 Description and Objectives


ProSIVIC is a 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 10: The ProSIVIC simulator

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

Pro-SiVIC™ allows to simulate custom scenarios involving road environment conditions, multiple sensors, dynamic actors and people to perform prototyping and testing stages through simulation.

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

2.2.2.2 Application domain and Use Cases

Pro-SiVIC is primarily dedicated to Automotive application (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™ 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 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 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 to modify the parameters influencing the behaviour of the sensors, of the environment and of the mobile objects.

2.2.2.3 Integration in the HF-RTP and interaction with other MTT

The various data streams generated by the Pro-SIVIC 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 is already fully connected with RTMaps (Fig. 9), and can be also connected with all other MTT through this INTEMPORA software.

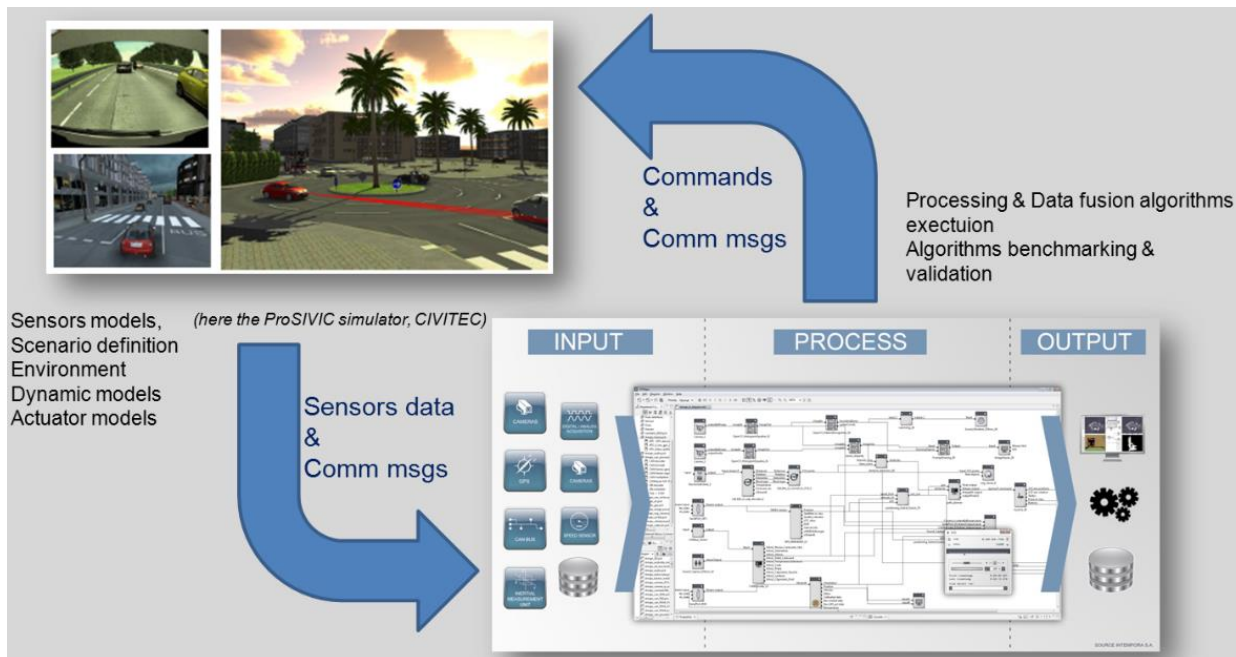




Figure 11: ProSIVIC interoperability with RTMaps

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

2.3 MTT for AdCoS design, verification and validation

Three main MTT for AdCoS design, evaluation and verification, provided by 3 HoliDes partners (BUT, ENA, UTO), have been currently identified to support WP4 objectives: the “AnaConDa + Race Detector & Healer + SearchBestie” tool chain, Djnn, and GreatSPN.

2.3.1 AnaConDa, Race Detector, Healer, SearchBestie (BUT)

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) 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 injections 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++.



HoliDes

Holistic Human Factors Design of
Adaptive Cooperative Human-
Machine Systems

hOLIDES

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 focus 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 12.

```
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");
}
```

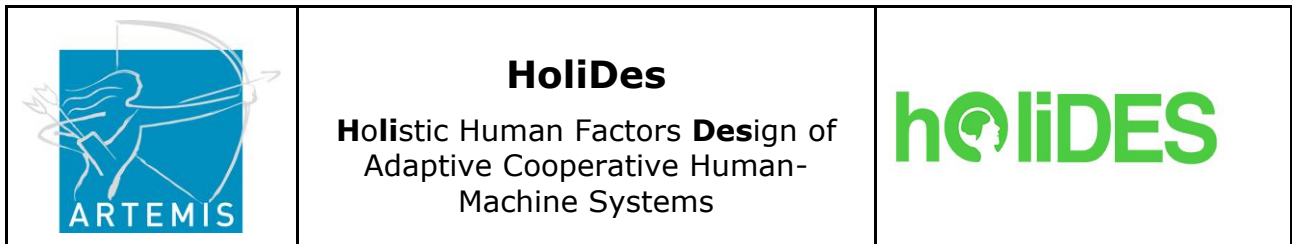


Figure 12: A simple analyser monitoring lock operations

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 13).

```
[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 13: An example of different noise settings for reads and writes.

The **Race Detector & Healer** is a prototype for a 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.

Race Detection & Healing tool can use either 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 14, 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 Race Detection & Healing 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 Detection & Healing tool use selected method to minimize the probability of context switch in the middle of the problematic atomicity section.

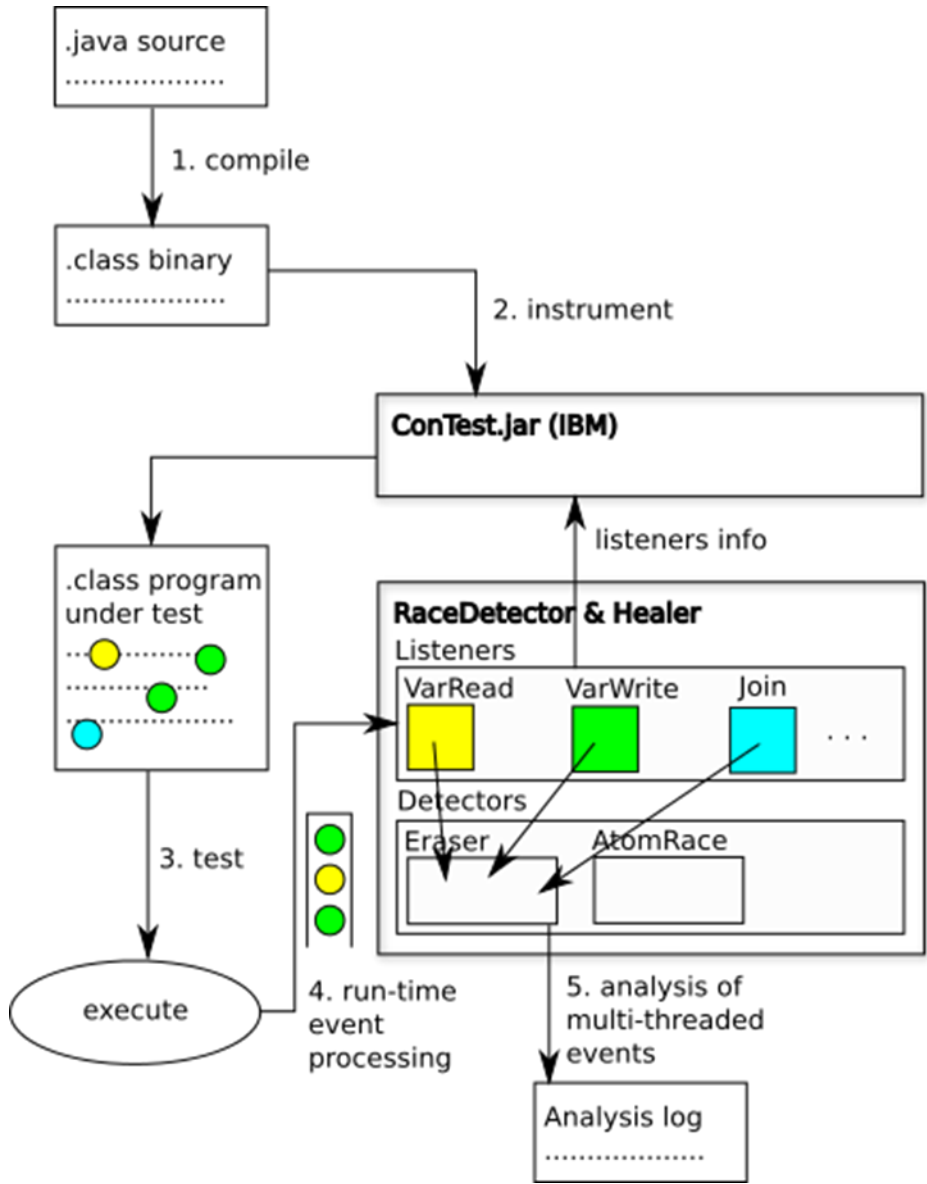




Figure 14: A scheme of usage RaceDetector & Healer when analysing concurrent Java programs.

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

SearchBestie (Search-Based Testing Environment) 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 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 (for WP7_HON_AER_REQ45: “The EFB System should be capable of alerting the flight crew of probable EFB system failures.”), system responsiveness / adaptation to internal workload (for WP7_HON_AER_REQ46: “The system should provide feedback to the user when user input is accepted. ...”), or verification of data integrity (for WP9_IFS_AUT_REQ03: “Data synchronization coming from different simulation tools (e.g. driver models, car sensors, road environment simulation, AdCoS, etc.) should be recorded in a synchronized way”).

2.3.1.3 Integration in the HF-RTP and interaction with other MTT

ANaConDa framework requires Intel's PIN framework to properly inject native code.

Race Detector & Healer works only in conjunction with instrumented Java programs and IBM ConTest which calls Race Detection & Healing tool at runtime.

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

2.3.2 Djnn (ENA)

2.3.2.1 Description and Objectives

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

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

A) Control primitives

Djnn relies on a fundamental control primitive called “binding”. A binding is a component that creates a coupling between two existing components. If there is a binding between components C1 and C2, then whenever C1 is activated, C2 is activated (C1 is called trigger and C2 is called action). A binding can be interpreted as a transfer of control, like a function call in functional programming or a callback in user interface programming.

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

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

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

Figure 15: Examples of bindings definitions in djnn

Bindings can be used to derive a set of control structures required to describe interactive software: Finite State Machine (FSM), Connector (used to transfer data between two components), Watcher (allow to connect C1 and C2 to C3 where C3 is activated only when C1 and C2 are synchronously

activated) or Switch (activates one of several components according to input data values).

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

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



Figure 16: examples of derived control structure definitions in djnn

FSM are one of the most used control structures for describing user interfaces with djnn. They contain other components named states and transitions. Transitions are bindings between two states (named origin and destination). A transition is active only when its origin is active. It behaves as a binding with a default action: changing the current state of the FSM to its destination state.

Therefore, the transitions define the inputs of the state machine: the state evolves on the sequence of activation of the triggers of the transitions, and ignores events that do not match the current state. Figure 17 shows the internal behaviour of a software button designed for use with a mouse: the djnn code at the left implements the FSM shown at the right. r is the image of the button (a rectangle component).

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

Figure 17: Example of a FSM definition

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

B) An architecture of reactive components

In djnn, every entity you can think of, abstract or physical, is a component. In addition to the control structures introduced above, djnn comes with a collection of basic types of components dedicated to user interfaces: graphical elements, input elements (mouse, multi-touch, sensors, etc.), file elements etc. Every type of component can be dynamically created or deleted.



To design interactive systems, components must be interconnected and organized. Interconnection is obtained with control structures, and can be performed independently of the nature and location of components. For example, a binding can connect the position of a mouse press to the position of a rectangle, so that the rectangle moves whenever the mouse is pressed. Structuration is obtained with a dedicated control structure: the parent-child interconnection that allows creating a hierarchy of components. For example, a complex graphical scene is composed of several graphical sub-components; a mouse is made of two buttons and one wheel; a FSM is made of several bindings etc. The designer can explicitly manage this tree-oriented architecture.

Combining the tree structure and the other control structures can be used for creating complex interactive behaviors and not only graphical scenes. For instance, combining FSMs by coupling their transitions, or by controlling the activation of one by a state or a transition of another, makes it possible to create complex behaviours (see example in Figure 18). The tree structure also makes it easier to structure applications as collections of reusable components.

```
# connect "trigger" transition to a component action "quit"
binding (mybutton/trigger, application/quit)
```

Figure 18: example of connection with FSM

Whenever a composite component is activated, the activation of its children components is iteratively performed. Each visited component is then activated and eventual transversal connections are activated.

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

2.3.2.2 Application domain and Use Cases

Djnn has been improved to prepare it for verification of interactive systems along two axes:

- Specification of the syntax and grammar through XML formats.
An abstract syntax and a grammar for djnn have been defined through a XML schema. The model addresses most components available in djnn, particularly control primitives.
- Development of a formal semantic in Petri nets.
Semantic of Djnn model is expressed through colored Petri nets (Jensen., 1996) extended with reset arc (Dufourd C., 1998). Composition of components is achieved through the merging of places of Petri nets.

These definitions are useful for verification according to 2 axes that are currently developed:



- static verifications based on the analysis of the abstract syntax of a djnn model
- dynamic verification based on the exploitation of the underlying Petri net model of the semantic.

2.3.2.3 Integration in the HF-RTP and interaction with other MTT

Djnn is a tool dedicated to design, verification and validation of user interfaces. This is why this framework can be interconnected with tools involved in the support of the engineering process for example:

- **MagicPED** for high level description of user interface: task model produced by MagicPED can be used as input for djnn
For this purpose, a mechanism of model transformation from MagicPED model to djnn model has to be defined and implemented.
- **GreatSPN**: model checker of GreatSPN can be used as a complementary way to perform formal verification on djnn model
For this purpose a model transformation between GreatSPN models and djnn model have to be defined and implemented.

Tools integration can be done through an OSLC based HF-RTP.

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

On another way, outputs produced by djnn are concrete user interfaces that can be used as part as an AdCoS: indeed several functionalities including adaptation of the user interface can be implemented in djnn.

In this case, interface produced by djnn must be integrated into an operational -real time- environment like RTMaps, CASCaS or COSMO-SIVIC.

2.3.3 GreatSPN (UTO)



2.3.3.1 Description and Objectives

GreatSPN (from UTO partner) is a tool with a common graphical interface for a set of formalism and solvers. The formalism considered are: Petri nets (Place/transition nets with priorities and inhibitor arcs), Colored Petri nets (the class called Symmetric Nets), their stochastic extensions to include random delays (GSPN and SWN) and Markov Decision Petri Nets (MDPN and the colored extension MDPN). The formalism of GreatSPN belong to the class of discrete events dynamic systems (DEDS).

The analysis for Petri nets and its colored extension 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.

GreatSPN includes also an innovative stochastic model checker, which allows to check CSLTA (Donatelli, 2009) properties, a superset of the well-known logic CSL (Aziz, 2000). GreatSPN allows also to solve 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.

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

When GSPN are extended with delays of general distributions, the tool provides a simulator.

2.3.3.2 Application domain and Use Cases



GreatSPN will be used in two different flavours in the project. The first one is the use of MDPN and MDP solution 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.

The second one is the use of GreatSPN has a verification/validation engine that will take two different directions: integration in the HF platform so as to work together with other Holidés tools and direct use of GreatSPN for the verification of requirements in the applicative WPs.

One of the HoliDes tools that have been identified for integration is Djnn (see 2.3.2): 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. We are currently investigating the possibility of using GreatSPN to provide a model checker for the task models tools of OFFIS.

For the applicative WP the work in WP2 (MTT_ requirement analysis excel file) has identified 35 requirements in which GreatSPN could be used. Of this 35, 28 are from automotive and will be accounted for in the specification of the WP9 MDP-based co-pilot, 2 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, 5 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.

For the time being UTO, in accordance with the HoliDes global plan, has given priority to the WP9 needs, while a more precise identification of the verification activities for GreatSPN is part of the second year effort.

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

2.3.3.3 Integration in the HF-RTP and interaction with other MTT

The tool solvers of MDP will be extended in the project to account for parameter uncertainty, incremental strategy computation and approximate strategy computation (to account for the short time available at run-time for the strategy computation). Since MDP will be used at run-time, the integration will be done inside RTMaps, as explained in Section 2.1.3.3

For the time being there has been no request from other partner to use the GreatSPN MDP solvers, so that there is no need to integrate them in the HF-RTP based on the OSLC paradigm.

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, it is instead at the moment not fully defined the tool chain and associated platform for the requirement validation of WP6 and WP8 using GreatSPN.



HoliDes

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

ho**l**i**D**E**S**

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

**They are only available in the
"Confidential Version" of D4.4.**



HoliDes

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

ho**l**i**D**E**S**

3 Integration Plan: HF-RTP tailoring in WP4

3.1 WP4 Strategy for integration plan

3.2 HF-RTP based on OSLC

3.2.1 A brief overview of OSLC

3.2.2 HF-RTP based on OSLC: Perspective of use for AdCoS

3.2.2.1 Requirements Tracing

3.2.2.2 Configuration Management

3.2.2.3 Reporting

3.2.3 Tool integration outside the scope of OSLC

3.3 HF-RTP based on RTMaps

3.3.1 RTMaps Tool chain for AdCoS design, development, simulation and evaluation

3.3.1.1 Connected tools

3.3.1.2 AdCoS development toolchain based on RTMaps

3.3.1.3 RTMaps for AdCoS verification and validation: IDEEP functions

3.3.2 Use of HF-RTP based on RTMaps for AdCoS design and evaluation: application to automotive domain

3.3.2.1 HF-RTP based on RTMaps for ADAS and AdCoS modelling

3.3.2.2 Main features to support AdCoS dynamic simulation

3.3.2.3 Toward a first tailored HF-RTP based on RTMaps for AdCoS verification and validation for automotive domain

3.3.2.4 Illustration of RTMaps application previously used for real and virtual ADAS/ADCOS design and test



HoliDes

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

ho**l**i**D**E**S**

4 WP4 MTT requirements updating

4.1 Requirements towards HF-RTP

- 4.1.1 COSMODRIVE and COSMO-SIVIC**
- 4.1.2 CASCaS**
- 4.1.3 MDP based Co-pilot**
- 4.1.4 GreatSPN**
- 4.1.5 Djnn**
- 4.1.6 AnaConDa, Race Detector, Healer and SearchBestie**

4.2 Requirements towards AdCoS

- 4.2.1 COSMODRIVE and COSMO-SIVIC**
- 4.2.2 CASCaS**
- 4.2.3 MDP based Co-pilot**
- 4.2.4 GreatSPN**
- 4.2.5 Djnn**
- 4.2.6 AnaConDa, Race Detector, Healer and SearchBestie**

4.3 Requirements towards other MTT

- 4.3.1 COSMODRIVE and COSMO-SIVIC**
- 4.3.2 CASCaS**
- 4.3.3 MDP based Co-pilot**
- 4.3.4 GreatSPN**
- 4.3.5 Djnn**
- 4.3.6 AnaConDa, Race Detector, Healer and SearchBestie**

5 Conclusion and perspectives





HoliDes

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

holiDES

6 References

- Aziz, A. S. (2000). Model-checking continuous-time Markov chains. . *ACM Trans. Comput. Log.* 1(1), 162–170.
- Beccuti M., F. 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., H. S. (2011). MDWNSolver: a framework to design and solve Markov Decision Petri Nets. (R. Consultants, Éd.) *International Journal of Performability Engineering*, 417-428.
- Bellet, T. B.-A. (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.
- Donatelli, S. H. (2009). Model checking timed and stochastic properties with CSLTA. *IEEE Trans. Software Eng.* 35(2), 224–240.
- Dufourd C., F. A. (1998). Reset nets between decidability and undecidability. *25th International Colloquium on Automata, Languages and Programming (ICALP'98)*, 1443, 103-115.
- Gruyer D., G. S. (2010). SiVIC, a virtual platform for ADAS and PADAS prototyping, test and evaluation. *FISITA'10*. Budapest.
- Gruyer D., G. S. (2011). Distributed Simulation Architecture for the Design of Cooperative ADAS. *FAST-ZERO (Future Active Safety Technology)*. Tokyo.

	<p style="text-align: center;">HoliDes Holistic Human Factors Design of Adaptive Cooperative Human- Machine Systems</p>	
--	--	---

Jensen., K. (1996). *Coloured Petri Nets*. Berlin: Heidelberg.

Kwiatkowska M., N. G. (2011). PRISM 4.0: Verification of Probabilistic Real-time Systems. *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)* (pp. 585-591). Springer.

Puterman, M. (2005). *Markov Decision Processes*. Chichester: Wiley.
Torino, U. d. (s.d.). www.di.unito.it/~greatspn. Récupéré sur GreatSPN website.