



HoliDes

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

D4.5 - Techniques and Tools for Model-based Analysis Vs1.5 incl. Handbooks and Requirements Analysis Update

Project Number:	332933
Classification:	Confidential Version
Work Package(s):	WP 4
Milestone:	M1
Document Version:	V1
Issue Date:	30.06.2015
Document Timescale:	Project Start Date: October 1, 2013
Start of the Document:	M15
Final version due:	M21
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 MTTs requirements updating, according to the progress done during the first phase of the project.
Compiled by:	N. González (ATOS), Pablo Nuño (ATOS)
Authors:	N. González (ATOS), B. Křena (BUT), A. Smrčka (BUT), M. Ferraton (CIV), T. Bellet (IFS), J.C. Bornard (IFS), D. Prun (ENA), S. Donatelli (UTO), JP. Osterloh (OFFIS)
Reviewers:	Martin Krähling (IBEO) Roberta Presta (Scienza Nuova)
Technical Approval:	Jens Gärtner, Airbus Group Innovations
Issue Authorisation:	Sebastian Feuerstack, OFFIS



HoliDes

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

RECORD OF REVISION

Date (DD.MM.JJ)	Status Description	Authors
31.03.2015	Initial version (Version.01)	N. González (ATOS)
23.04.2015	Anaconda, SearchBestie, RaceDetector & Healer MTTs handbooks	B. Křena (BUT), A. Smrčka (BUT)
24.04.2015	Pro-SIVIC handbook	M. Ferraton (CIV)
30.04.2015	Aesthetic changes	N. González (ATOS)
13.05.2015	COSMODRIVE and COSMO-SIVIC contributions	Thierry Bellet (IFFSTAR) Jean Charles Bornard (IFFSTAR)
22.05.2015	Contributions	ENAC, ITEMPORA, BUT
03.06.2015	Contributions about MDP and GreatSPN	Susanna Donatelli (UTO)
04.06.2015	Contributions about CASCaS	Jan Patrick Ostelroth (OFFIS)
15.06.2015	Start of the internal quality process	Martin Krähling (IBEO) Roberta Presta (Scienza Nuova)
28.06.2015	Compile partners answers to reviewers comments	Pablo Nuño (ATOS)



Table of Contents

- 1 Introduction: WP4 objectives and MTTs..... 9**
- 2 Handbook of Techniques and Tools for Model-based Analysis..... 13**
 - 2.1 Human Operator Models 13
 - 2.1.1 COSMODRIVE (IFFSTAR)..... 13
 - 2.1.1.1 Model Description and Objectives..... 13
 - 2.1.1.2 Application domain and Use Cases 14
 - 2.1.1.3 Integration in the HF-RTP and interaction with other MTTs .16
 - 2.1.2 CASCaS (OFFIS) 20
 - 2.1.2.1 Description and Objectives..... 20
 - 2.1.2.2 Application domain and Use Cases 22
 - 2.1.2.3 Integration in the HF-RTP and interaction with other MTTs .22
 - 2.1.3 MDP-based Co-pilot (University of Torino) 23
 - 2.1.3.1 Application domain and Use Cases 24
 - 2.1.3.2 Integration with HF-RTP and interaction with other MTTs.... 25
 - 2.2 MTTs specifically dedicated to support AdCoS Modelling and their Virtual Simulation 28
 - 2.2.1 RTMaps (INTEMPORA) 28
 - 2.2.1.1 Description and Objectives..... 28
 - 2.2.1.2 Application domain and Use Cases 29
 - 2.2.1.3 Integration in the HF-RTP and interaction with other MTTs .30
 - 2.2.2 Pro-SIVIC (CIVITEC) 31
 - 2.2.2.1 Description and Objectives..... 31
 - 2.2.2.2 Application domain and Use Cases 32
 - 2.2.2.3 Integration in the HF-RTP and interaction with other MTTs .33
 - 2.3 MTTs for AdCoS design, Verification and Validation 35
 - 2.3.1 AnaConDa, Race Detector & Healer, SearchBestie (Brno University of Technology) 35
 - 2.3.1.1 Description and Objectives..... 35
 - 2.3.1.2 Application domain and Use Cases 39
 - 2.3.1.3 Integration in the HF-RTP and interaction with other MTTs .39
 - 2.3.2 djnn (ENAC)..... 40
 - 2.3.2.1 Description and Objectives..... 40
 - 2.3.2.1.1 Detail of extension 1.1: Static analysis of djnn models ..42



HoliDes

Holistic Human Factors Design of
Adaptive Cooperative Human-
Machine Systems



2.3.2.1.2	Extension 1.2: Translation of djnn model to Petri net model	46
2.3.2.1.3	Details of extension 2: djnn for dynamic analysis	48
2.3.2.2	Application domain and Use Cases	51
2.3.2.3	Integration in the HF-RTP and interaction with other MTTs	51
2.3.3	GreatSPN (University of Torino)	52
2.3.3.1	Description and Objectives	52
2.3.3.2	Application domain and Use Cases	53
2.3.3.3	Integration in the HF-RTP and interaction with other MTTs	55
3	Integration Plan: HF-RTP tailoring in WP4	58
3.1	WP4 Strategy for integration plan	58
3.2	HF-RTP based on OSLC	58
3.2.1	A brief overview of OSLC	58
3.2.2	HF-RTP based on OSLC: Perspective of use for AdCoS	58
3.2.2.1	Requirements Tracing	58
3.2.2.2	Configuration Management	58
3.2.2.3	Reporting	58
3.2.3	Tool integration outside the scope of OSLC	58
3.3	HF-RTP based on RTMaps	58
3.3.1	DDS – Distributed Data Services	58
3.3.2	RTMaps Tool chain for AdCoS design, development, simulation and evaluation	58
3.3.2.1	Connected tools	58
3.3.2.2	AdCoS development toolchain based on RTMaps	58
3.3.2.3	RTMaps for AdCoS Verification and Validation: The IDEEP framework	58
3.3.3	Use of HF-RTP based on RTMaps for AdCoS design and evaluation: application to automotive domain	58
3.3.3.1	HF-RTP based on RTMaps for ADAS and AdCoS modelling	58
3.3.3.2	Toward a first tailored HF-RTP based on RTMaps for AdCoS Verification and Validation for automotive domain	58
4	WP4 MTTs requirements updating	58
4.1	Requirements towards HF-RTP	58
4.1.1	COSMODRIVE and COSMO-SIVIC	58
4.1.2	CASCaS	59
4.1.3	MDP based Co-pilot	59



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



- 4.1.4 GreatSPN.....59
- 4.1.5 djnn.....59
- 4.1.6 AnaConDa, Race Detector & Healer and SearchBestie59
- 4.2 Requirements towards AdCoS.....59
 - 4.2.1 COSMODRIVE and COSMO-SIVIC59
 - 4.2.2 CASCaS.....59
 - 4.2.3 MDP based Co-pilot.....59
 - 4.2.4 GreatSPN.....59
 - 4.2.5 djnn.....59
 - 4.2.6 AnaConDa, Race Detector & Healer and SearchBestie59
- 4.3 Requirements towards other MTTs.....59
 - 4.3.1 COSMODRIVE and COSMO-SIVIC59
 - 4.3.2 CASCaS.....59
 - 4.3.3 MDP based Co-pilot.....59
 - 4.3.4 GreatSPN.....59
 - 4.3.5 djnn.....59
 - 4.3.6 AnaConDa, Race Detector & Healer and SearchBestie59
- 5 Conclusion and perspectives.....59**
- 6 References.....60**

List of figures

Figure 1: WP4 MTTs for AdCoS Design, Verification and Validation.....	11
Figure 2: Functional architecture of the AdCoS based on MOVIDA.....	15
Figure 3: IFFSTAR use of Pro-SiVIC and RTMaps for ADAS modelling and simulation	16
Figure 4: RTMaps diagram for Pro-SiVIC/COSMODRIVE simulation	17
Figure 5: Pro-SiVIC simulation with COSMODRIVE at the wheel	18
Figure 6: Overview of the V-HCD platform, as an example of a tailored HF-RTP based on RTMaps for automotive application	18
Figure 7: V-Design process of AdCoS with the COSMODRIVE platform.....	19
Figure 8: Structure of the cognitive architecture CASCaS with all the internal components and the major data flows.	21
Figure 9: GreatSPN GUI while playing the token game.....	26
Figure 10: Integration of MDP in RTMaps	27
Figure 11: The RTMaps Studio.....	29
Figure 12: An AdCoS development toolchain	31
Figure 13: The Pro-SiVIC [®] simulator.....	32
Figure 14: Pro-SiVIC [®] interoperability with RTMaps	34
Figure 15: A simple analyser monitoring lock operations	36
Figure 16: An example of different noise settings for reads and writes.	37
Figure 17: A scheme of usage Race Detector & Healer when analysing concurrent Java programs.	38
Figure 18: djnn platform architecture	41
Figure 19: djnn binding component expressed in Petri net.....	46
Figure 20: djnn assignment component expressed in Petri net.....	47
Figure 21: Approach for verification	48
Figure 22: djnn applications inputs/outputs.....	49
Figure 23: djnn RTMaps integration	50
Figure 24: The use of the GreatSPN model-checking facilities for the application WPs.....	55



HoliDes

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

HoliDes

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 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 a first example of tailored HF-RTP interconnecting WP4 MTTs, in order to support AdCoS validation and verification. Section 4 is dedicated to MTTs requirements updating, according to the progress done during the first phase of the project. Lastly, Section 5 introduces the next steps of WP4 for the next phases of the project.

1 Introduction: WP4 objectives and MTTs

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**² (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

² “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).



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 Techniques and Tools (MTTs) 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 MTTs 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 Techniques 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).

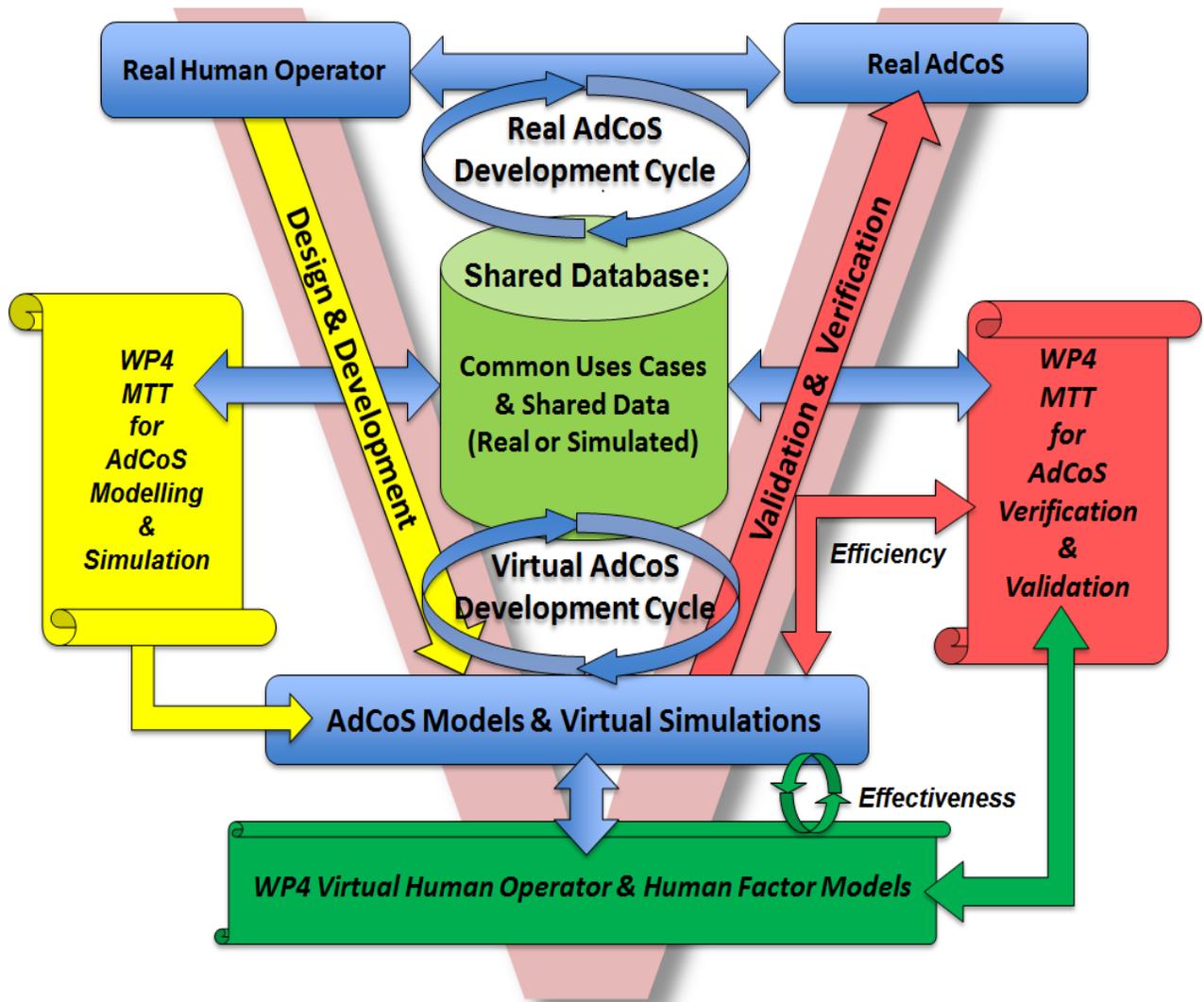


Figure 1: WP4 MTTs for AdCoS Design, Verification and Validation

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



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

Currently, 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) MTTs specifically dedicated to support AdCoS Modelling and their Virtual Simulation,
- 3) MTTs 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.

2 Handbook of Techniques and Tools for Model-based Analysis

2.1 Human Operator Models

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

2.1.1 COSMODRIVE (IFFSTAR)

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. Through this 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.
- Simulate two 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* process (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 the 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

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

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 D2.5) 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 challenge 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 (presented in the following Figure and further described in D3.5).

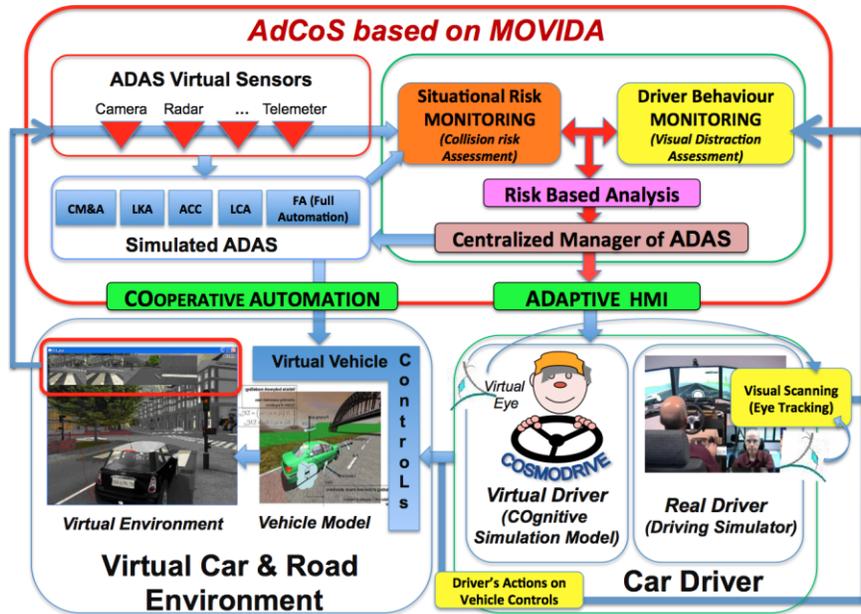


Figure 2: Functional architecture of the AdCoS based on MOVIDA

To support this global objective, COSMODRIVE have been interfaced in WP4 with the Pro-SiVIC platform, as used in IFSTTAR-SIVIC (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.

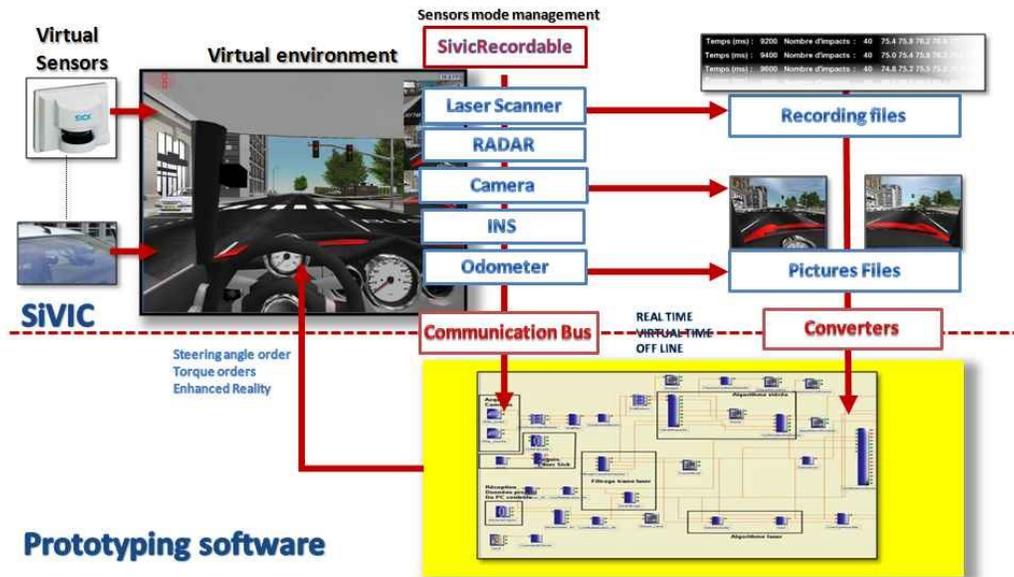


Figure 3: IFFSTAR use of Pro-SiVIC and RTMaps for ADAS modelling and simulation

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 AdCoS design and evaluation, COSMODRIVE has been connected in WP4 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 through SiVIC, Pro-SiVIC and RTMaps) into (4) a virtual 3D road environment (simulated with Pro-SiVIC).

RTMaps offers an easy way to connect multiple tools and data sources. For the HoliDes project, multiple partners already use it. COSMODRIVE can be interfaced with RTMaps, and RTMaps is planned to be the way to interact with COSMODRIVE for the HoliDes project. Components are currently under development, but the main ones already are working, which allows the basic communications.



HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

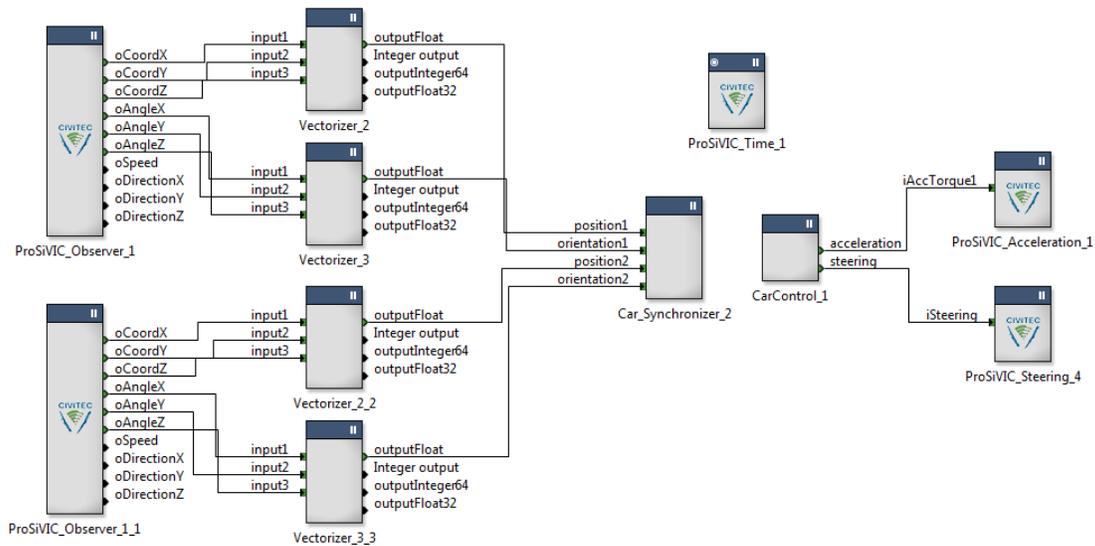


Figure 4: RTMaps diagram for Pro-SiVIC/COSMODRIVE simulation

The RTMaps diagram presented in the figure above provides an overview of the COSMODRIVE integration/interfacing with this software (and then, with all other MTTs connected with RTMaps). The diagram presented in the left part of the figure is responsible for getting both cars' position and orientation, and for sending them to COSMODRIVE by means of the Car_Synchronized component. Thereby, COSMODRIVE will then synchronize its duplicate of the world. The right part shows a COSMODRIVE component providing "Car Control" (acceleration and steering) to Pro-SiVIC.

In WP4 and WP9, the challenge is to use this COSMODRIVE-based 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). To support this final objective, the first step was to give to COSMODRIVE the possibility to dynamically drive a virtual Pro-SiVIC car in the Pro-SiVIC 3D environment. From the joint effort done by IFFSTAR, INTEMPORA and CIVITEC, all these MTTs were interconnected to support such dynamic simulation. Figure 5 shows an example of the results of that interconnection by representing a Pro-SiVIC simulation with COSMODRIVE at the wheel.



HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

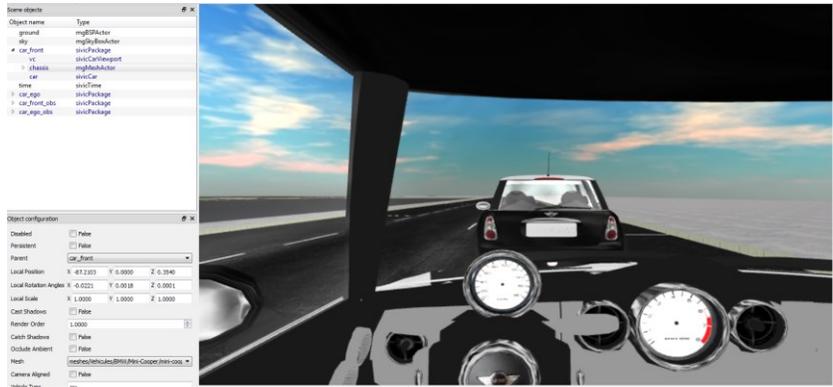


Figure 5: Pro-SiVIC simulation with COSMODRIVE at the wheel

In addition, and in a more general way, all the MTTs to be used for supporting the MOVIDA-AdCoS design process are now integrated in the V-HCD platform by means of the RTMaps software. The following figure provides an overview of this V-HCD based on RTMaps, for supporting the MOVIDA-AdCoS virtual design and evaluation, and then, its potential transfert towards real cars, by means of some RTMaps functionalities.

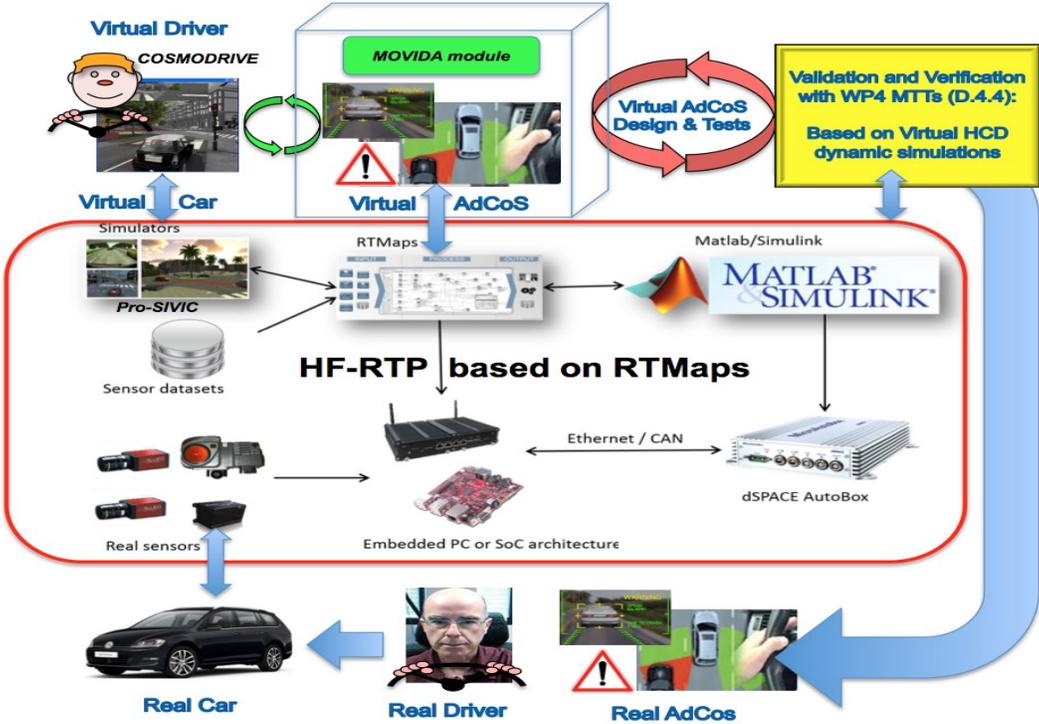


Figure 6: Overview of the V-HCD platform, as an example of a tailored HF-RTP based on RTMaps for automotive application



In this approach, RTMaps, Pro-SiVIC and the V-HCD platform will support virtual simulations of car sensors, ADAS and AdCoS, to be used by driver models (COSMODRIVE). Simulated data, or data collected on real car (e.g., a WP9 demonstrator) could be also used to design MOVIDA-AdCoS algorithms. Connection with other MTTs in WP4, like djnn or GreatSPN, could 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 could also provide a support for the transfer of virtual AdCoS toward real cars and/or for empirical data sharing with IAS, CRF and TAK demonstrators.

The following figure presents an overview of the V-HCD platform use to support the virtual design, prototyping and evaluation of future AdCoS (here based on MOVIDA). In this “V design process”, only the steps integrated under the red line will be effectively implemented by IFFSTAR during the HoliDes project, with the aim of validating and demonstrating 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 IFFSTAR).

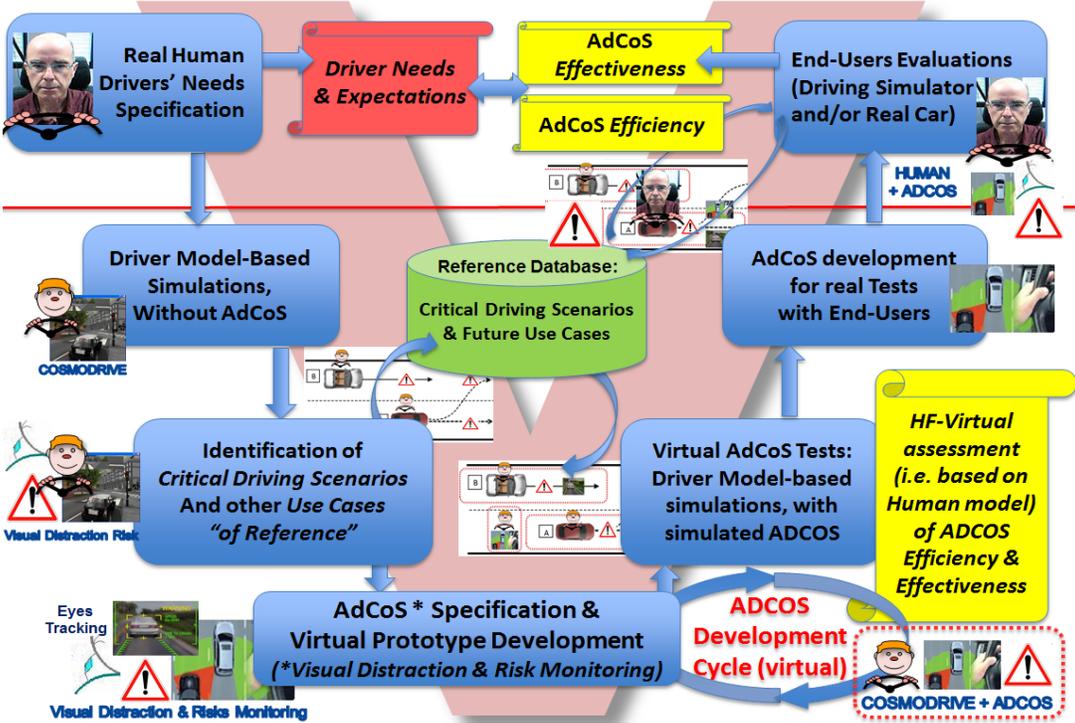


Figure 7: V-Design process of AdCoS with the COSMODRIVE platform



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 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 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 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". Then, during the virtual design process of the MOVIDA-AdCoS, this reference database associated with visual distraction simulations based on COSMODRIVE will be used to progressively increase the MOVIDA-AdCoS *efficiency* (i.e., AdCoS developments & virtual tests Cycle on the figure) in accordance with the different variations of the critical scenarios previously identified. Such COSMODRIVE + AdCoS based simulations will also allow the designer (i) to assess the potential *effectiveness* of MOVIDA before developing a real prototype, and (ii) to test its effectiveness with real human drivers, through full scale tests with end-users implemented on driving simulators and/or with real cars (final stage of the design process).

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.



HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

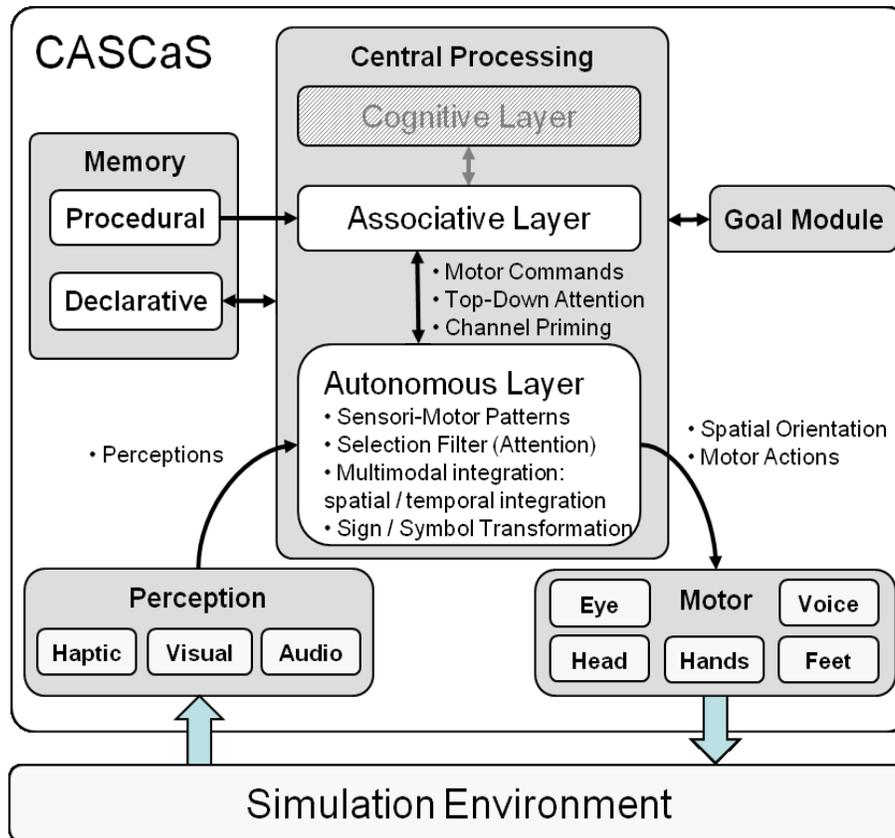


Figure 8: Structure of the cognitive architecture CASCaS with all the internal components and the major data flows.

Figure 8 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 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

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

/ 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. Details of this are still under discussion, and the concrete work on this is planned for the end of the second year.

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). HF-RTP integration of MagicPED, which is described in D2.5, has been started.

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

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 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

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.

To account for uncertainty MDPN have been extended to MDPNU (MDPN with uncertainty) through a 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, in which probabilities are specified through a lower and an upper Bound) 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.

2.1.3.1 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.

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

2.1.3.2 Integration with HF-RTP and interaction with other MTTs

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.

Integration with GreatSPN.

The GreatSPN suite of University of Torino provides a framework to design and solve MDPN models by means of specific modules [Beccuti 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 around 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 9 shows the GreatSPN GUI while playing the token game: the possible events (transition firings) are enlighten (right box), while the boxes on the left allow observing the history of the animation.



HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

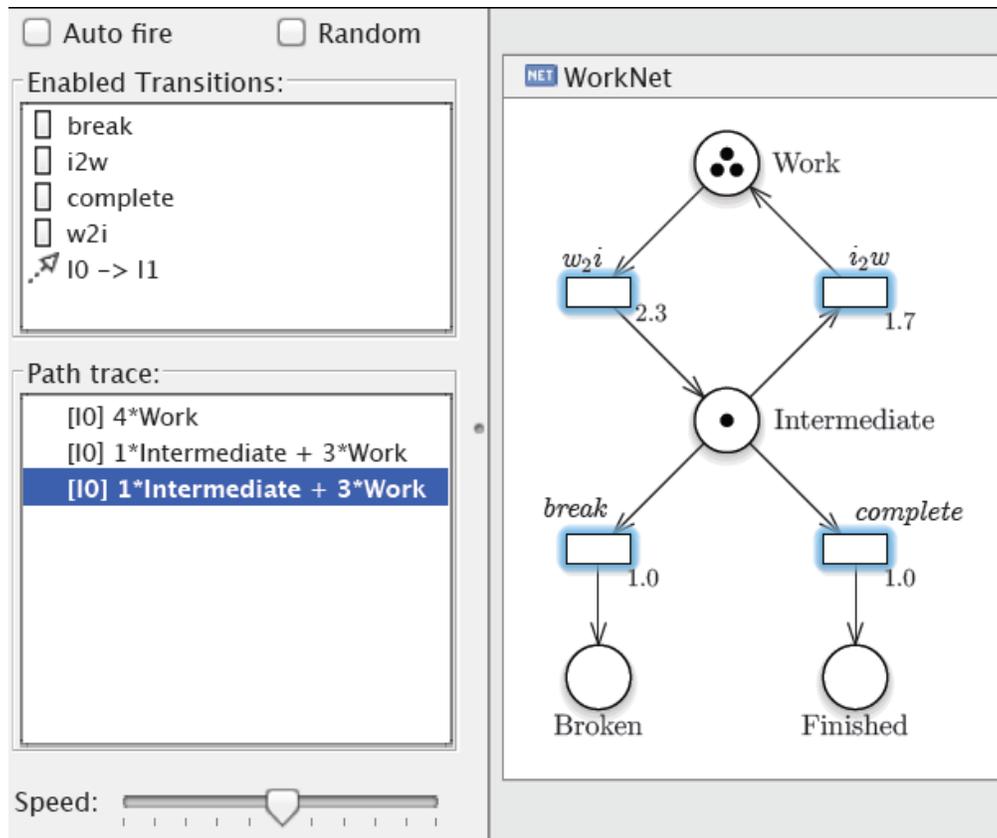


Figure 9: GreatSPN GUI while playing the token game

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.

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 an 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 10 shows the structure of the RTMaps component that contains the MDP solver of GreatSPN.

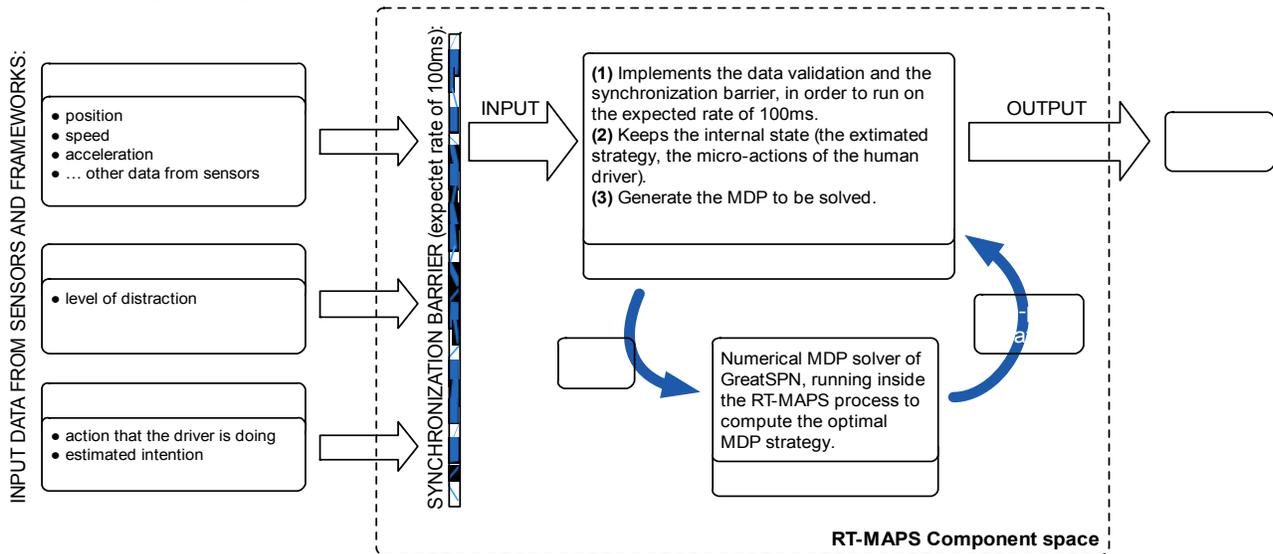


Figure 10: Integration of MDP in RTMaps

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, which 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

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

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). The traces are currently being collected through the experimental campaign of CRF. If the results of this validation steps are not sufficient to gain the right level of confidence we shall consider the integration with Pro-SIVIC ([2.2.2](#)) platform for the generation of specific scenarios to experiment with different traces.

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

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 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

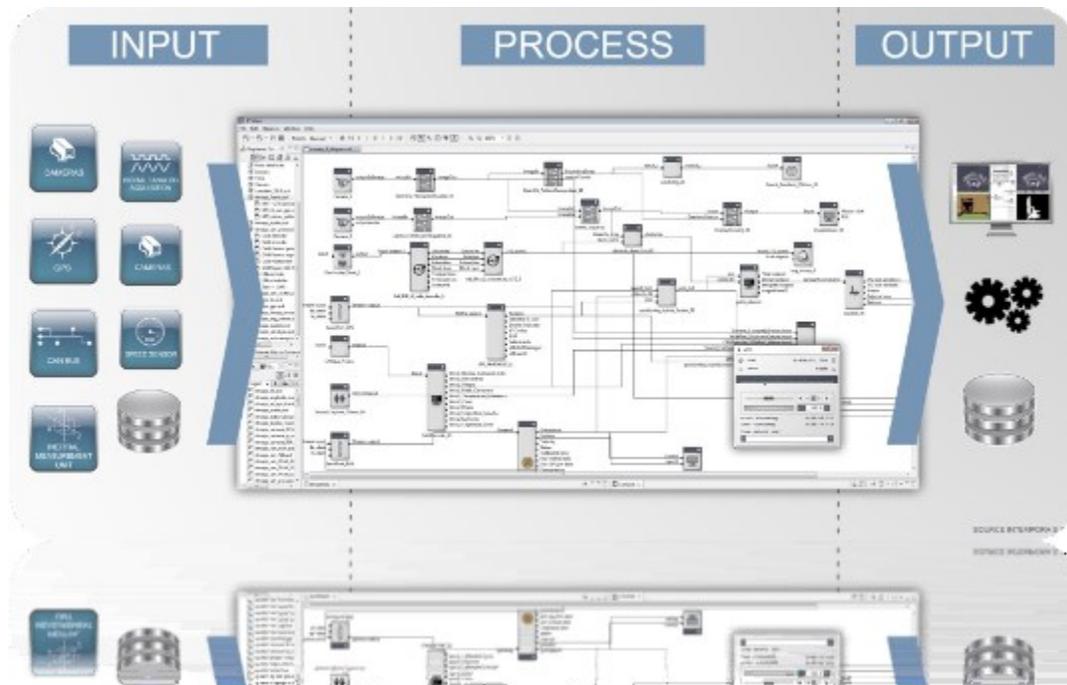


Figure 11: The RTMaps Studio

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

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

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 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

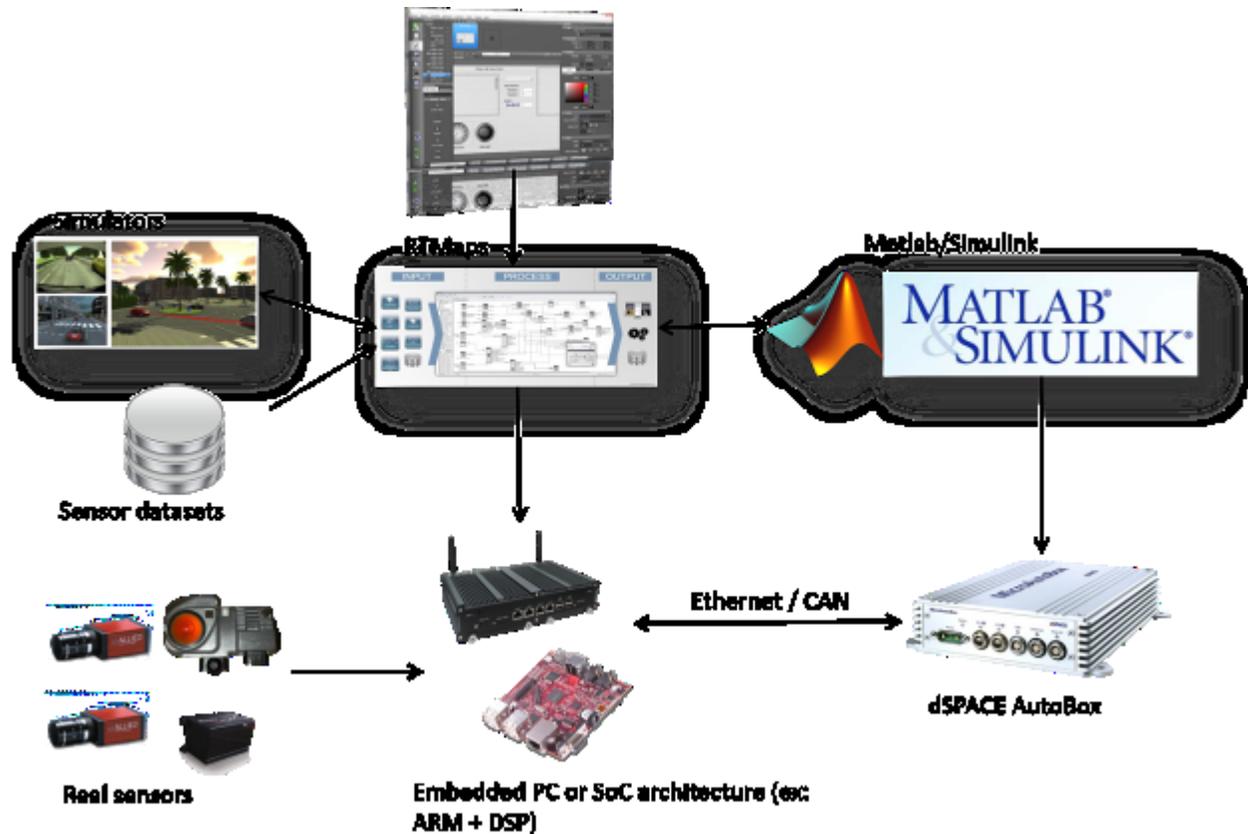


Figure 12: An AdCoS development toolchain

2.2.2 Pro-SIVIC (CIVITEC)

2.2.2.1 Description and Objectives

Pro-SIVIC[®] 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.



HoliDes

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

HoliDes



Figure 13: The Pro-SIVIC[®] simulator

Pro-SIVIC[®] 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 simulating custom scenarios involving road environment conditions, multiple sensors, dynamic actors and people to perform prototyping and testing stages through simulation.

2.2.2.2 Application domain and Use Cases

Pro-SIVIC[®] 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[®] 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 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[®] 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 MTTs through this INTEMPORA software.

	<p>HoliDes</p> <p>Holistic Human Factors Design 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) 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++.

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 15.

```

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 15: 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 16).

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

The **Race Detector & Healer** 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 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 17, 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.

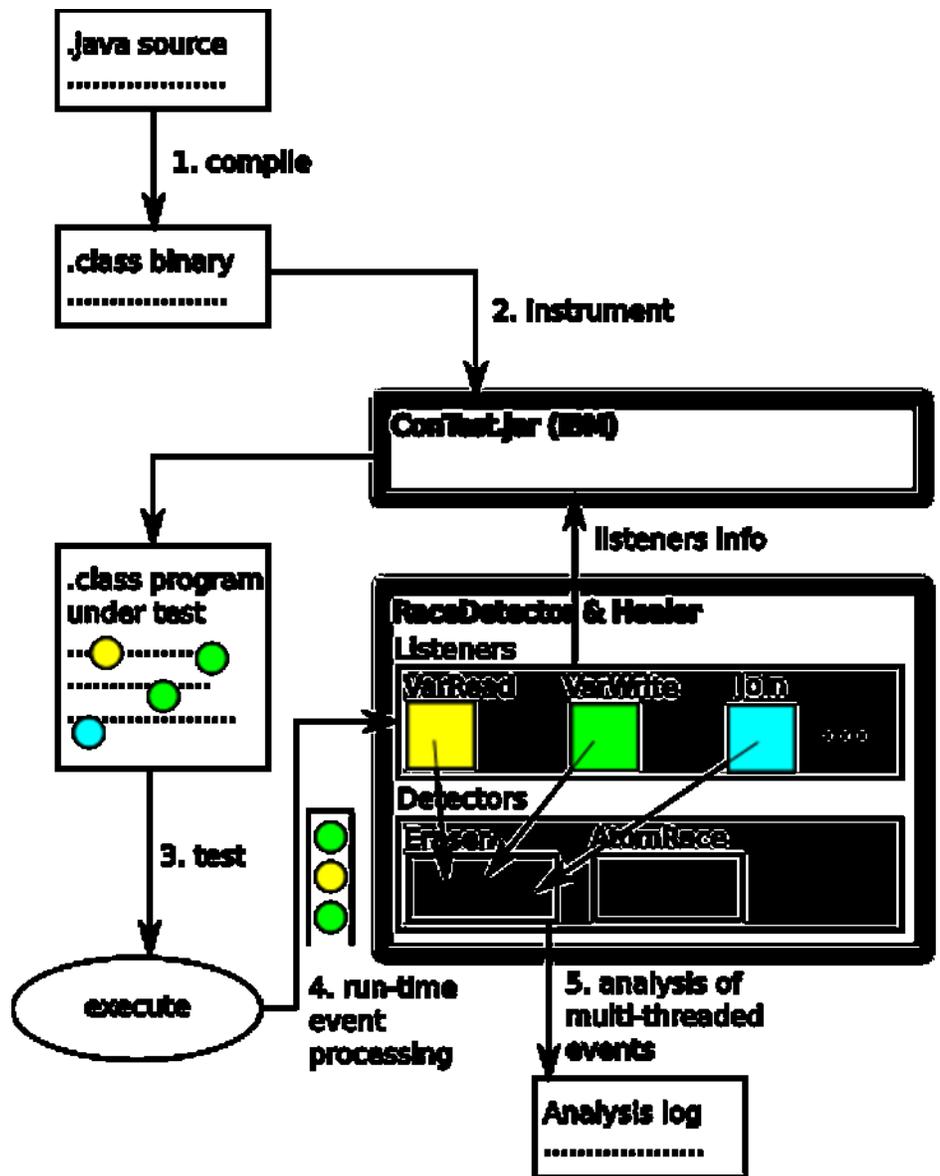


Figure 17: A scheme of usage Race Detector & Healer when analysing concurrent Java programs.

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).

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

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].

There is an interest in using Race Detector & Healer tool within WP6 – Health use cases expressed recently by Carlos Cavero Barca from ATOS. We are also working with Honeywell to move from declared interest to real usage of ANaConDA within WP7. In particular, they have already learned the framework and are applying it on their programs.

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

ANaConDa framework and its analysers focus on 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.

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

2.3.2 djnn (ENAC)

2.3.2.1 Description and Objectives

djnn

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

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

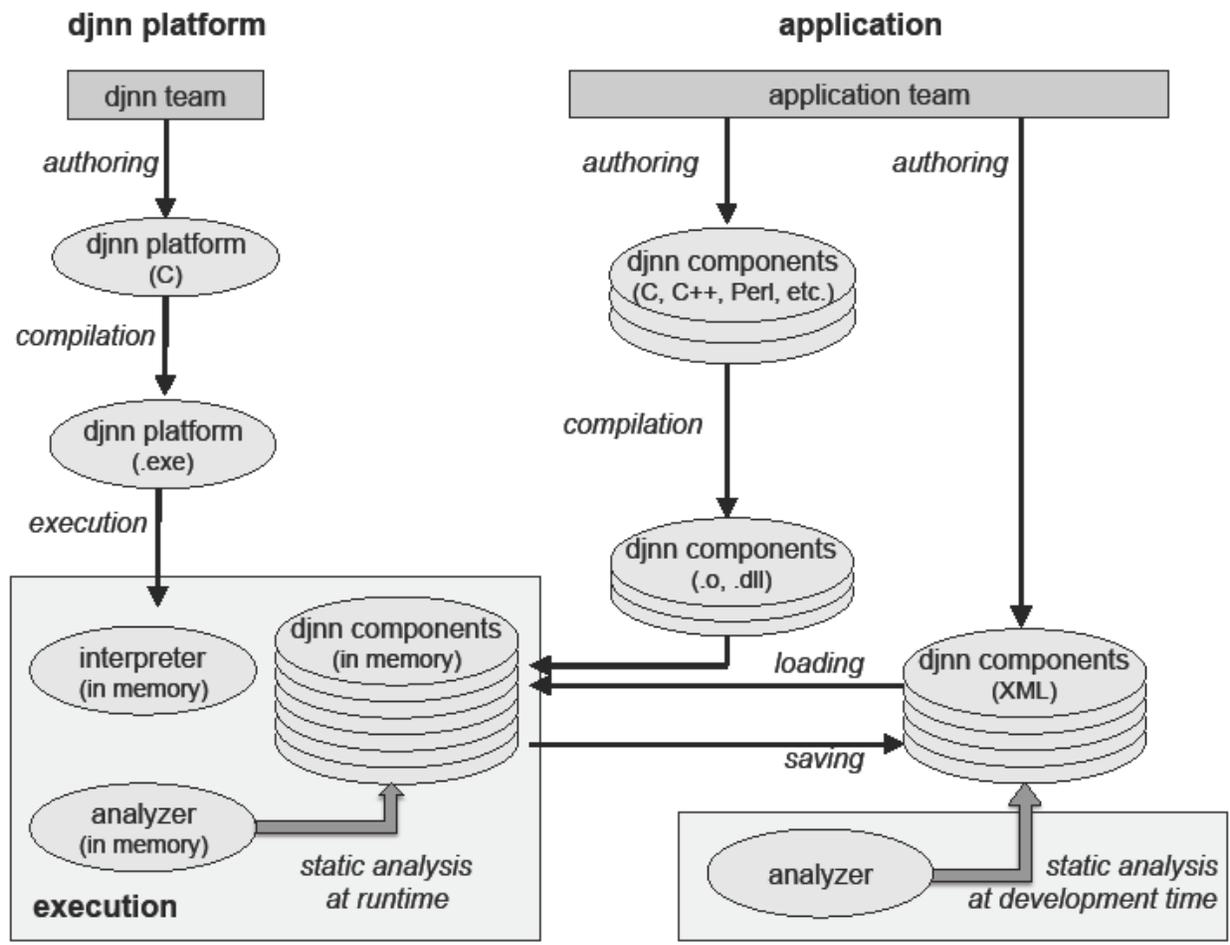


Figure 18: djnn platform architecture

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.

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. They are included in a specific platform dedicated to the simulation of AdCoS.

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

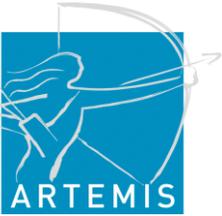
2.3.2.1.1 Detail of extension 1.1: Static analysis of djnn models

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.

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.

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

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.

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,

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

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 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.

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

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:

- 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.

2.3.2.1.2 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.

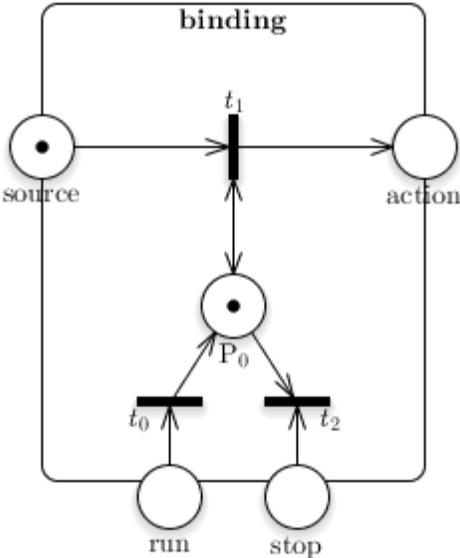


Figure 19: djnn binding component expressed in Petri net



HoliDes

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

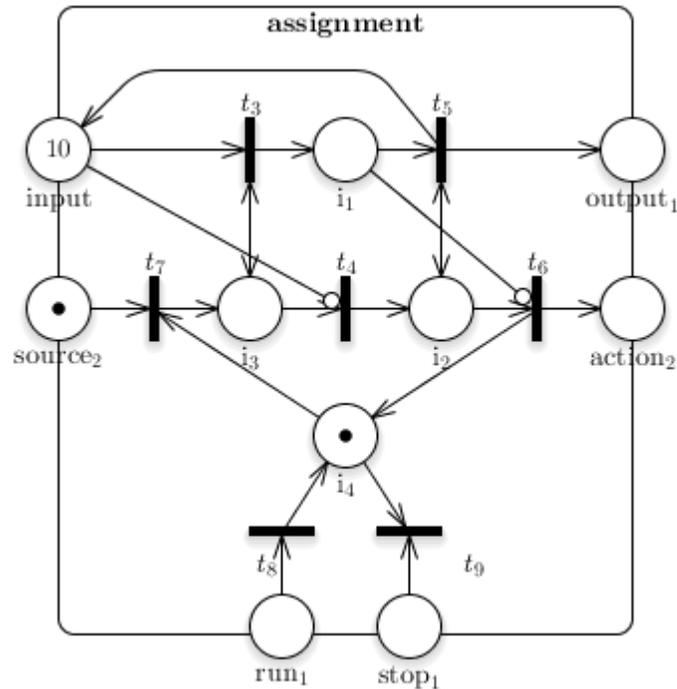


Figure 20: djnn assignment component expressed in Petri net

Composition of components is expressed asynchronously through place merging.

Example: the figure below show 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.



HoliDes

Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems

HoliDes

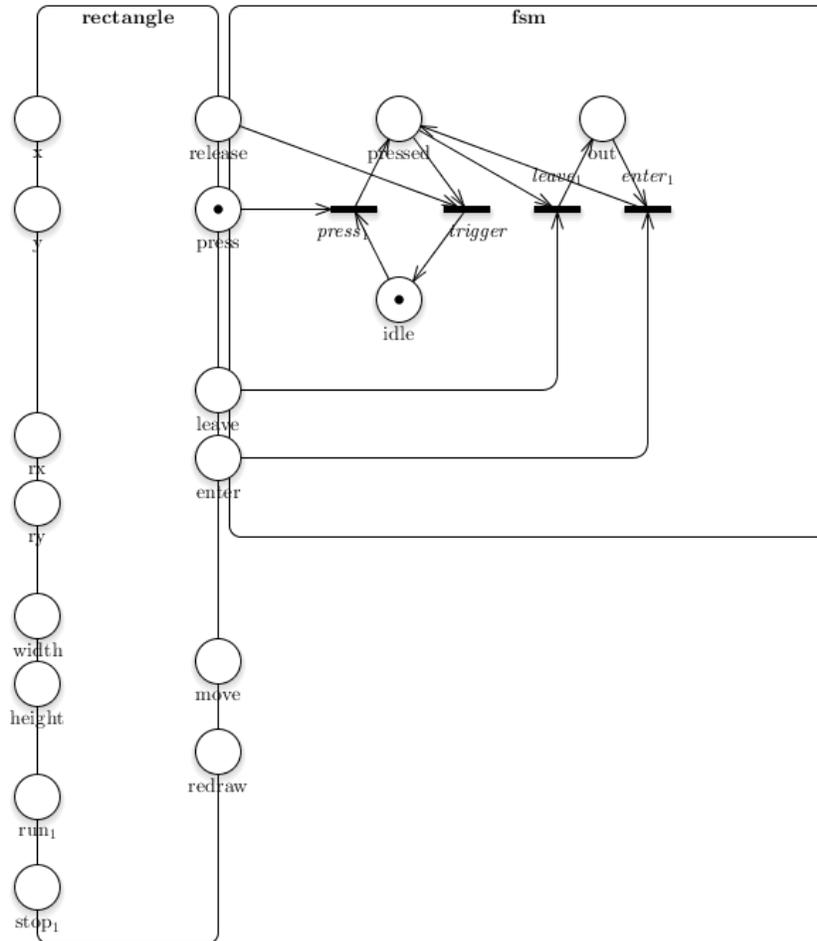


Figure 21: Approach for verification

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.

2.3.2.1.3 Details of extension 2: djnn for dynamic analysis

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 is currently being extended as follows:

- Integration with the RTMaps platform. We are currently exploring two ways for achieving that integration. Firstly djnn interpreter can be directly integrated in RTMaps through RTMaps components.

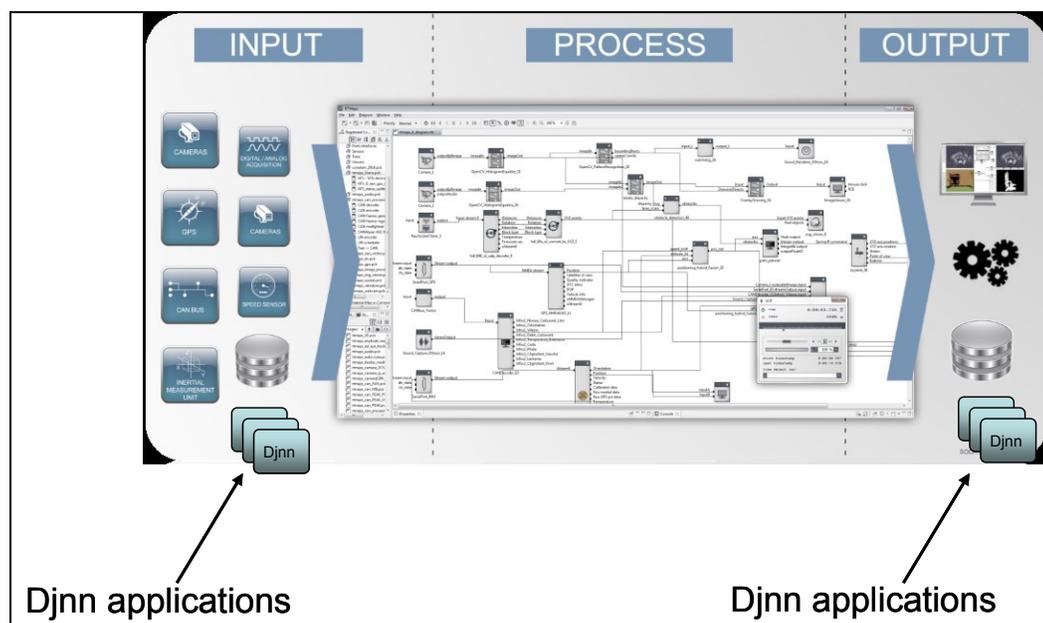


Figure 22: djnn applications inputs/outputs

Secondly, djnn and RTMaps can be interconnected through a communication layer:

	<p>HoliDes</p> <p>Holistic Human Factors Design 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: Static analysis of djnn models: properties are checked 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). 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 for Petri net description (PNML : Petri Net Markup Language)
- Exchange of models through OSLC

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

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.

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)

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

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 will be 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_requirement 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, 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.

In accordance with the HoliDes global plan, priority was given on the first year to WP9 needs, but we are currently supporting IREN, ReLab and EADS in the identification 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 24 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.

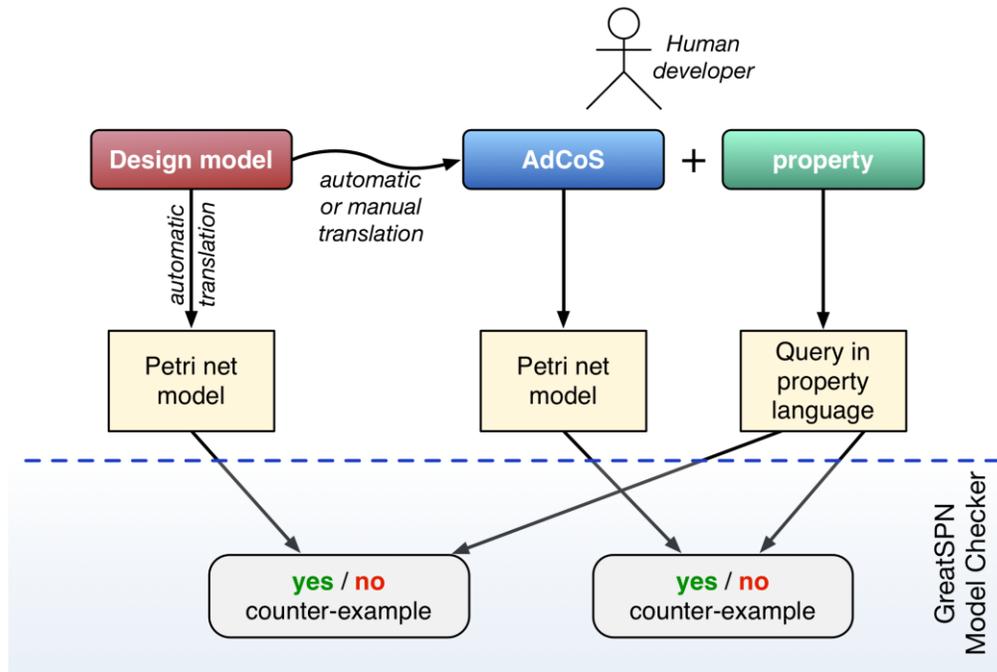


Figure 24: The use of the GreatSPN model-checking facilities for the application WPs

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

Also for what concerns integration we have two different flavours of integration, one for the MDP part of GreatSPN and one for the model checking part.

The tool solvers of MDP have been extended in the project to account for parameter uncertainty and will be extended to include 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 2.1.3.2.

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.

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

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 with other MTTs.

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

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

**They are only available in the
"Confidential Version" of D4.5.**



HoliDes

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

HoliDes

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 DDS – Distributed Data Services

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

3.3.2.1 Connected tools

3.3.2.2 AdCoS development toolchain based on RTMaps

3.3.2.3 RTMaps for AdCoS Verification and Validation: The IDEEP framework

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

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

3.3.3.2 Toward a first tailored HF-RTP based on RTMaps for AdCoS Verification and Validation for automotive domain

4 WP4 MTTs requirements updating

4.1 Requirements towards HF-RTP

4.1.1 COSMODRIVE and COSMO-SIVIC



HoliDes

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

HoliDes

- 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 MTTs

- 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.
- Bause, F et al (1995) Abstract Petri Net Notation, Petri Net Newsletter, No 49 (1995) 9-27.
- 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.
- 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, 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.
- 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.

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

- 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, U. d. (s.d.). www.di.unito.it/~greatspn. Récupéré sur GreatSPN website.
- 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