HoliDes

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

D4.1 - Requirements Analysis for Model Based Analysis Techniques and Tools

Project Number:	332933
Classification:	Public
Work Package(s):	WP4
Milestone:	M1
Document Version:	V1.0
Issue Date:	10.04.2014
Document Timescale:	Project Start Date: October 1, 2013
Start of the Document:	Month 05
Final version due:	Month 06
Deliverable Overview:	**Main document:** Holides-WP4-D4_1-v1.0.doc (Public) **Annex I**: Holides-WP4-D4_1-v1.0.xlsx (Public)
Keywords:	Requirement Analysis, Model Based Analysis
Compiled by:	Daniel Prun (ENA)
Authors:	Daniel Prun (ENA) Jan-Patrick Osterloh (OFF) Bohuslav Křena (BUT) Stefan Rieger (TWT) Zdenek Moravek (HON)
Technical Approval:	Jens Gärtner, Airbus Group Innovations
Issue Authorisation:	Sebastian Feuerstack, OFFIS

RECORD OF REVISION		
Date	Status Description	Author
17.03.2014	Initial version	Daniel Prun (ENA)
02.04.2014	Compilations of additions after WP4 internal reviews from : - Jan-Patrick Osterloh (OFF) - Bohuslav Křena (BUT) - Stefan Rieger (TWT) - Zdenek Moravek (HON)	Daniel Prun (ENA)
08.04.2014	Compilation of update after WP4 external reviews from : - Fabio Tango (CRF) - Lars Weber (OFF) - Mark Eilers (OFF)	Daniel Prun (ENA)
10.04.2014	Addition of few modifications from: - Thierry Bellet (IFS)	Daniel Prun (ENA)

Table of Contents

List of figures

Acronyms

UC - Use Case
UML - Unified Modelling Language

1 Introduction

This document describes the objectives, the methodology and the results of the analysis of requirements which have been collected from application work packages (WP6 to WP9) regarding their relevance for model based analysis techniques and tools.

Even if only one release is planned by the HoliDes project plan, this document must be considered as a "living document". Indeed, according to feedbacks from the first iterations of the project, considering further possible developments of supporting tools and foreseen numerous evolutions of requirements, this document will have to be updated accordingly.

Inputs:
Inputs used to produce this document are:

- Requirements from WP6-WP9
  - D6.1: Health related scenario descriptions – Vs 1.1 – 15/02/2014
  - D7.1: Requirements Definition for the HF-RTP, Methodology and Techniques and Tools from a Aeronautics Perspective – Vs 1.0 – 12/02/2014
  - D8.1: Requirements Definition for the HF-RTP, Methodology and Techniques and Tools from a Control Room Perspective – Vs 0.8 – 14/02/2014
  - D9.1: Requirements Definition for the HF-RTP, Methodology and Techniques and Tools from an Automotive Perspective – Vs 0.1 – 14/02/2014
- List of tools: Tool listing V4 (file "HoliDes - Tools-listing.xlsx") dated 17/03/2014

Outputs:
This document contains:

- A selection of requirements that will be analysed during WP4 activities. Criteria used for the selection are identified in chapter 3.
- The identification of techniques and tools to be used in WP4 for the analysis of selected requirements (from WP6-9).

- Improvement suggestions regarding requirements formulation in order to improve quality and usability for verification and validation activities (main criteria are introduced in chapter 3.3).

Life-cycle:
This document is an important input for numerous WP4 tasks and will be used to produce:

- D4.2: Plan for Integration of Model-based Analysis Techniques and Tools into the HF-RTP and Methodology.
- D4.3: Metrics for Model-based and Empirical AdCoS Qualification.
- D4.4, D4.5, D4.6 and D4.7: Techniques and Tools for Model-based Analysis Vs1.0, Vs 1.5, Vs 1.8 and Vs 2.0 incl. Handbooks and Requirements Analysis Update.

This document is intended to be used as input for the first release of HF-RTP 0.5 (month 08). For cycle 2 and cycle 3 it will be updated according to:

- first feedback from application on WP6-9 applications
- internal WP4 feedback
- use of new tools
- use of updated tools
- modification of input requirements from WP6-WP9

2 Background and context

According to the objectives stated in the HoliDes description of work, the global objective 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*".

In this chapter, we give background related to this objective.

2.1 Model based analysis

Verification and validation are two system engineering technical processes (ISO IEC 2008). Their objectives consist of:

- **Verification**: confirmation, through the provision of objective evidence, that specified requirements have been fulfilled.

- **Validation**: confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled.

Verification focuses on technical requirements coming from the engineering point of view (and not from the user point of view). In other words, verification tries to answer to the following question "Are we building the system right?").
On the contrary, target of validation deals with final user and operational related requirements, trying to answer to "Are we building the right system?", or in other words:"Are we building the system fulfilling the user needs?".

Model-based analysis is an approach to support verification and validation processes. The idea is to construct an intermediate representation of the future system – the model - and to search for evidences directly on this representation. With this approach, evidence can be a mathematical demonstration or a global observation performed on all possible states of the system.

Compared to other V&V methods (for example: testing performed on final products), the main advantage of model-based analysis lies in the fact that it can be performed very early in system development, before the implementation phase. According to several studies (Baziuk 95, Boehm 76), this allows to considerably reduce the cost of bug detection and correction.

2.2 Techniques

In the context of HoliDes project, four different techniques for model based analysis will be considered: model checking (section 2.2.1), evaluation (section 2.2.2), theorem proving (section 2.2.3) and abstract interpretation (section 2.2.4).

2.2.1 Model checking

Basically, the approach of model checking relies on three steps which are illustrated in Figure 1:

- Step 1: From an informal specification of system requirements (alternatively from an existing system analysis), a model of the future

system is built. Properties to be checked on the system are also expressed through a formal model (through a logic formula).

- Step 2: Based on the system simulation, states are explored and for each of them the validity of properties is checked.
- Step 3: After partial or exhaustive coverage of system states (according to the kind of properties to be checked), validity of properties can possibly be resolved. They can be true, false (then a counter example can be showed) or unresolved.

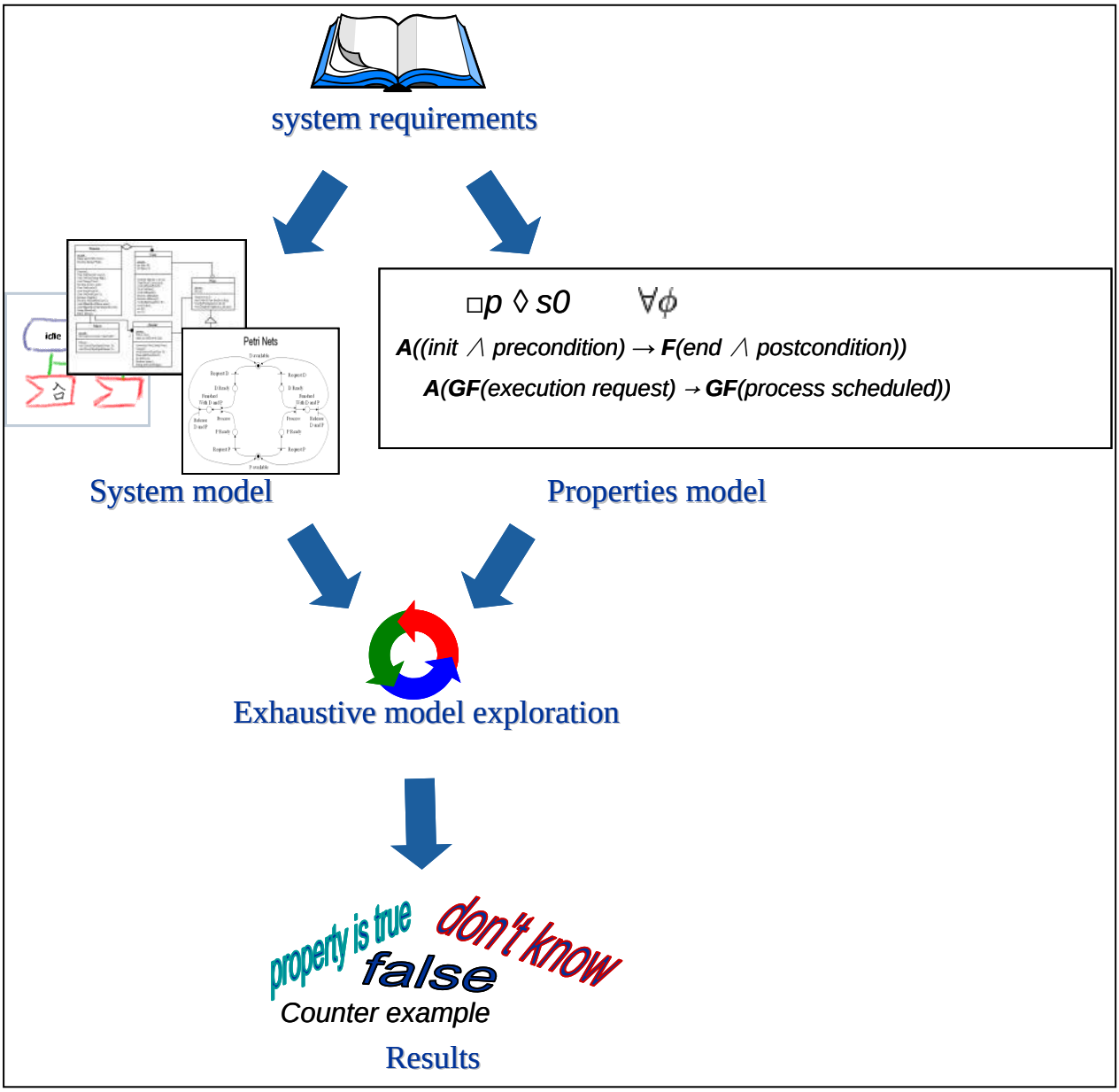

Figure 1: Model checking technique

The model checking approach allows to find potential bugs related to internal consistency of system requirement (by checking model and properties consistency) and to potential information added into the model, which were not part of the initial requirement specification (this is achieved by checking whether added information do not violate - or conflict - initial system

requirements). Added information generally includes design related information like functional and logical architecture.

A strong limitation of model checking lies in the fact that for large systems, an exhaustive model check (i.e. an analysis able to explore all states of the system) is practically impossible, due to the huge (potentially infinite) number of states to address during analysis. Therefore, a lot of model checking simulation techniques have been developed with the objective to soften this limit: e.g. by using optimized data structures (such as BDDs: Binary Decision Diagrams which allowed to address system up to 10^{100} states), development of new simulation techniques (for example symbolic model checker) or development of model transformations that reduce the number of states without modifying properties validity etc.

2.2.2 Evaluation[1]

In contrast to the previously explained model checking approach, the evaluation does not transform the informal description into a formal system specification. Instead, a prototype is produced and various simulators are used in order to play (to simulate) missing parts and the environment. Again, we have three main steps presented in Figure 2:

- Step 1: from an informal specification, a prototype of the future system is built. Additional tools may be used in order to simulate different parts and components (for example tools dedicated to simulate the behaviour of a car driver, tools to simulate weather extreme conditions for an aircraft etc…). Specific Verification and Validation tools (called "observers") are defined in order to produce measures that will be used to check properties.
- Step 2: Prototypes are executed and observations are performed. During execution, observers produce raw data reflecting the measure to be performed.
- Step 3: Raw data is analysed and conclusions on properties can be drawn.

[1] This approach is often called «simulation» which is ambiguous because model checking approach relies also on simulation (of model). In order to avoid this confusion, we decided to keep the term "evaluation", used here as synonym of "activity of testing a prototype".

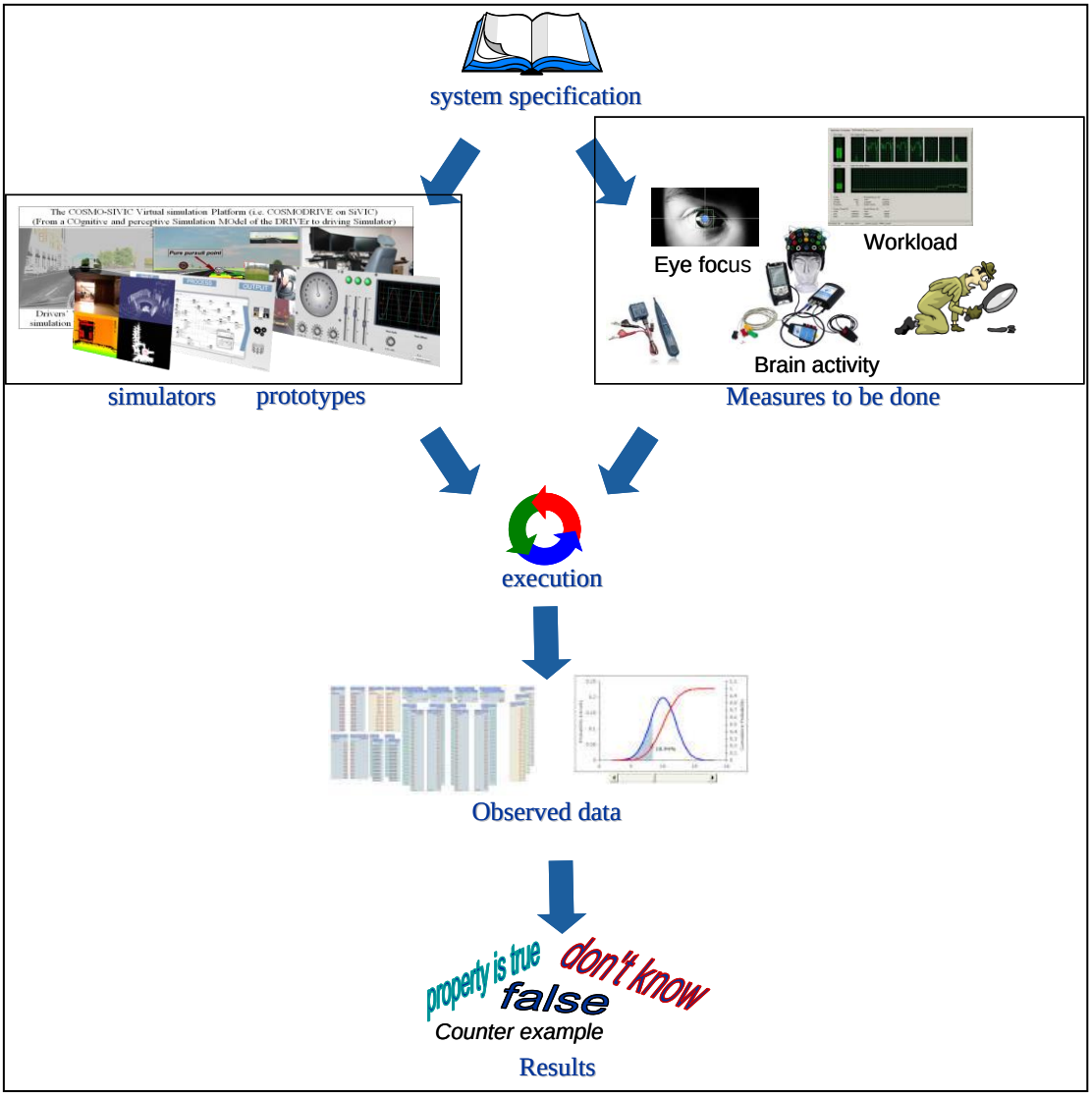

Figure 2: Evaluation technique

2.2.3 Theorem proving

This approach is similar to model checking. But instead of model simulation mechanisms, a *proof mechanism* is used. By applying various deduction rules, a proof assistant allows to perform mathematical demonstration of properties on the model.

For example, the natural deduction system consists of a set of rules of inference for deriving consequences from premises. A proof tree is built in such a way the root is the proposition to be proved and the leaves are the initial assumptions or axioms. Proof trees are usually drawn with the root at the bottom and the leaves at the top.
For example, one rule is known as modus ponens which specified that if *we know P is true, and we know that P implies Q, then we can conclude Q*.

$$\frac{P \quad P \Rightarrow Q}{Q}$$ (modus ponens)

Both the premises and the conclusion may contain meta-variables (in this case, P and Q) representing arbitrary propositions. When an inference rule is used as part of a proof, the meta-variables are replaced in a consistent way with the appropriate kind of object (in this case, propositions). Most rules come in one of two flavours: introduction rules introduce the use of a logical operator while elimination rules eliminate it. The previously explained modus ponens is an elimination rule for the logical operator "⇒" .

In the natural deduction system, several other rules do exist (a total of fifteen rules for natural deduction system) and can be combined. In the example below, nine rules are combined to prove that : $(A \Rightarrow B \Rightarrow C) \Rightarrow (A \wedge B \Rightarrow C)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{[y : A \wedge B]}\,(\mathrm{A})}{A}\,(\wedge\mathrm{E}) \quad \dfrac{}{[x : A \Rightarrow B \Rightarrow C]}\,(\mathrm{A})}{B \Rightarrow C}\,(\Rightarrow\mathrm{E}) \quad \dfrac{}{[y : A \wedge B]}\,(\mathrm{A})}{B}\,(\wedge\mathrm{E})}{C}\,(\Rightarrow\mathrm{E})}{A \wedge B \Rightarrow C}\,(\Rightarrow\mathrm{I,y})}{(A \Rightarrow B \Rightarrow C) \Rightarrow (A \wedge B \Rightarrow C)}\,(\Rightarrow\mathrm{I,x})$$

The main advantage of this approach lies in the fact that it allows to encompass systems with a potentially infinite number of states: as in mathematics where a proof can be performed on infinite sets of elements, theorem proving is applicable on systems with an infinite number of states.

2.2.4 Abstract interpretation

Creating a model of a system in order to verify the system under investigation is time consuming and, in addition, can be a source of errors (spurious errors can be introduced into the model during the modelling while some real errors can be hidden, for instance, by the model abstraction). In later phases of software development when source code and/or executable application is available, other approaches to system verification may be considered. Actually, the system itself can be understood as its own model or a model can be inferred from the system automatically. Such approaches are exploited in model checkers for common programming languages like Java (Visser, Havelund, Brat, Park, & Lerda, 2003) or C (Henzinger, Jhala, Majumdar, & Sutre, 2003; Clarke, Kroening, Sharygina, & Yorav, 2005) as well as in various kinds of static analysis (Kam & Ullman, 1976; Cousot & Cousot, 1992; Nielson & Nielson, 1999).
Another issue for verification is concurrent software that is now very popular due to wide usage of multi-core processors and multi-processors. Verification of concurrent systems is more difficult due to huge number of possible process interleaving. It causes that precise approaches like model checking do not scale well while common testing methods are not able to reveal rarely manifesting bugs. In order to increase chances to spot these errors, techniques such as injection of noise into the scheduling of concurrent processes have been proposed and supported by tools like IBM ConTest (Edelstein, Farchi, Nir, Ratsaby, & Ur, 2002) or ANaConDA (Fiedor & Vojnar, 2012). Furthermore, various dynamic analyses have been proposed. They are used to analyse the behaviour seen in a testing run in order to detect concurrency bugs like data races or deadlocks.

2.3 Tools

Several tools do support techniques introduced above. We briefly present those tools that are likely to be used for the first HF-RTP release.

2.3.1 Tools description

GreatSPN
GreatSPN is provided by University of Turin - Department of Computer Science. It is intended to be used for model checking.

GreatSPN is a suite of tools for modelling, validation, optimization, and performance evaluation of complex systems using Generalized Stochastic Petri Nets and their extensions such as, for instance, Stochastic Well-formed Nets and Markov Decision Petri Nets. It provides a user friendly framework to experiment with stochastic Petri net based modelling techniques and thanks to the implementation of efficient analysis algorithms it can be used also to study real complex applications.

RTMAPS + ProCIVIC + COSMO-CIVIC
These tools are introduced in the same chapter because we plan to use them jointly in order to support design, development and evaluation techniques of AdCOS. While RTMAPS and ProCIVIC can be used independently of the application domain (equally on WP6, WP7, WP8 and WP9 application domains), COSMO-CIVIC is dedicated to automotive system domain.

In particular, **RTMaps** is provided by INTEMPORA. It is intended to be used to support design and development phases of AdCOS, and specifically for WP4 their verification and validation through the evaluation technique.
RTMaps is a rapid and modular development environment for real-time multi-sensor (more generally multi-I/O) applications. It allows to very easily acquire and process data from various data sources such as cameras, audio, eye-trackers, biometric sensors, motion capture, CAN bus, GPS, IMUs, Lidars, Radar etc. It provides data samples timestamping functionalities and allows real-time recording and playback of the data for post-analysis, self confrontation, and so on. It provides a graphical environment for rapid development based on existing components, and a C++ SDK for integration of third-party libraries into components.

ProSIVIC is provided by CIVITEC and is intended to support, in the context of WP4, the evaluation technique.
ProSIVIC is a modelling and simulation software for 3D environments and multi-frequency sensors such as cameras, Lidars, Radars, IMUs, GPS etc. It helps designing and validating applications from the early development stages. It is oriented towards embedded systems with perception capabilities, with or without human interaction in the simulation.

COSMO-CIVIC is provided by IFSTTAR and is intended to support, in the context of WP4, the evaluation technique.

COSMO-SIVIC is a simulation research tool designed during the ISI-PADAS project (2008-2011), integrating a driver model (named COSMODRIVE for COgnitive Simulation MOdel of the DRIVEr) able to drive a virtual car within a virtual environment (based on a SiVIC precommercial version of ProSIVIC). During the HoliDes project, we plan to interface this research tool with ProSIVIC and RTMaps, in order to support virtual simulation of future AdCos use by human drivers (simulated by COSMODRIVE) and then to support a "Human Centered Design approach" of Cooperative driving Aids in WP9. Moreover, COSMO-SIVIC could be also used as a driving simulator, for implementing experiments and tests among real human drivers.

CoSimECS

It is provided by OFFIS and is intended to support the evaluation technique.
CoSimECS is a tool, allowing the specification of a

- system in terms of agents, tasks and resources
- simulation in terms of assigning simulators for the agents and resources, as well as automated generation of configuration files

CoSimECS also allows setting up and controlling the simulation, based on the OFFIS simulation platform. The OFFIS simulation platform is based on the High Level Architecture standard (IEEE1516). Development of CoSimECS has been started in D3CoS, and will be continued in HoliDes. Currently it has been only used internally at OFFIS, but when tool maturity allows, a release to the OFFIS partners using CASCaS for evaluation and simulation is planned.

CASCaS

It is provided by OFFIS and is intended to support the evaluation technique.
CASCaS is a cognitive architecture, intended to allow simulation of human behaviour, based on psychological and physiological sound models of human behaviour. When connected to a simulator, CASCaS performs actions and made decisions by applying a procedures model based on the current status of the simulation.

AnaConDa, Race Detector & Healer and SearchBestie

These tools are provided by VeriFIT research group from Brno University of Technology and are intended to support checking of concurrent software.
ANaConDA is a framework that simplifies the creation of dynamic analysers for analysing multi-threaded C/C++ programs on the binary level. The Java Race Detector & Healer is a prototype for a run-time detection and healing of

data races and atomicity violations in 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).

2.3.2 Tools classification

Table below shows which techniques are supported by which tools:

	Model checking	Evaluation	Theorem proving	Abstract interpretation
GreatSPN				
RTMAPS + ProSivic + COSMO-CIVIC				
CoSimECS				
CASCaS				
AnaConDa, Race Detector & Healer SearchBestie				

Theorem proving technique is covered by no selected tool. This implies that the first HF-RTP release will not propose support for theorem proving. For this release, effort will be done on development of model checking and evaluation techniques. However, for the following releases of HF-RTP, more effort could be done on the development of theorem proving technique through search and adaptation of existing free methods (like B-Method) and tools (like RODIN or B-Toolkit tools).

3 Properties relevant for WP4

Requirements collected from application work-packages (WP6-WP9) are not necessarily suitable for model based verification and validation. The objective of this chapter is to characterise requirements which are relevant for model-

based analysis techniques and tools. For this purpose, we use three groups of criteria:

- **target**: must describe
  - adaptation aspects of Human-Machine systems or
  - tools integrated in RTP that should be used to support WP4 analysis
- **origin**: must take under consideration requirements from applications as well as requirements from norms and standards
- **quality**: requirements must be usable for verification and/or validation purpose

The next chapters detail these groups of criteria.

3.1 Properties are related to adaptation

The objective of WP4 is to provide methods and tools to allow verification of properties related to adaptation.

Cooperative Human-Machine Systems encompass systems where many humans and many machines are inherently cooperating to achieve some common, superordinate goals or tasks. We are beyond a one-to-one static and directional relationship between human (considered as master of the relation) and machine (considered as slave). We consider systems where interactions between humans and machines are complex: multiple, dynamic, in constant evolution, hard to predict.

Adaptation is a central point of interest of cooperative systems. It can be defined as the capability of a system to adapt itself to changes occurring in its external context (e.g. weather, traffic or environment) as well as in its internal perimeter (e.g. status of the human operators or current tasks).

For our analysis, we will consider two targets for properties related to adaptation: cause and effect.

- The "**cause**" properties are focused on the decision mechanism leading to trigger an adaptation mechanism. They are based on measuring, interpreting and/or predicting the internal and external contexts of the system in order to decide of a dedicated adaptation mechanism. For example: to observe the current status of a human operator or the automated parts of the system.

More precisely, we will consider properties related to the triggers of an adaptation mechanism (i.e.: what is at the origin of an adaptation):

- Modification of human characteristics:
  - Modification of the psycho-physical status of the human such as workload, time pressure, distraction, emotional modification.
  - Modification of the background of the human such as language, geographical localization, experience.
- Modification of machine characteristics (e.g.: failure, dysfunction)
- Modification of the environment:
  - Evolution of the volume of submitted requests such as number of incoming data/events to be treated, number of interfaces to manage.
  - Evolution of the quality of the interface between the system and its environment (e.g.: throughout, quality)
- Modification due to an internal decision

- The "**effect**" properties consider properties related to the modification mechanisms themselves, e.g., global warning, alarm set off, or global system self-reconfiguration.
  - Internal modifications:
    - Task and/or resources re-allocation
    - Re-organisation of the system architecture: introduction (or suppression) of agents, resources, interfaces.
  - External modifications:
    - Warning, alarms to the operators
    - Modification of the expected functionalities and associated performances

3.2 Properties imposed by regulations

Development and qualification of AdCoS in safety critical domains like Transportation, Control Rooms and Health has to comply with human factors and safety regulations:

- IEC 62366 for, Health;
- JAA TGL36, ISO/IEC12207, ARINC 661, EU-OPS1, EU-FCL Subpart Q for Aeronautics;
- ISO-Standard 924, ZDv91/11, IEC EN 61508 for Control Rooms;
- ISO 26262, RESPONSE Code of Practise for Automotive.

Properties imposed by these regulations are:

- the human operator can override the machine agents at all times
- machine agents take full control only if human operators are no longer capable to guarantee safe operation
- human operators always detect that adaptations have occurred and why
- the machine agents adapt their interaction to their human operators, e.g. in terms of languages, preferences, level of education, usual behaviour
- adaptation leads to safe, robust, resilient, efficient and effective overall system behaviour.

3.3 Quality of requirements for verification and/or validation

Requirements are expected to have the following properties to be suitable for V&V purposes:

- **Consistent**: requirements should be consistent in the sense that they must be free of internal contradictions. We cannot accept a property expressed to be true in a certain context by a requirement and expressed to be false in the same context by another requirement.
- **Measurable**: it should be possible to determine if a property is true or false, independently of all human interpretation or judgement.
- **Unambiguous**: requirement should be subject to one and only one interpretation. Vague subjects, adjectives, prepositions, verbs and subjective phrases should be avoided in order to lead to an objective interpretation.

4 Selected requirements

Requirements from application work-packages (WP6 to WP9) have been analysed and, on the basis of relevant properties as defined in chapter 3, some of them have been selected for model based verification and validation. Selected requirements will be the basis for further WP4 work: models, methods, tools will be developed to perform verification and validation of these requirements.

Selected requirements are listed below. Refer to Annex 1 for a full description.

- WP6_ATO_HEA_REQ25 The system SHOULD be operational in case of failures
- WP6_AWI_HEA_REQ01 The operator model should be able to identify the operators' skill and experience level through their (overt) actions.
- WP6_AWI_HEA_REQ03 The human factors models should allow a simulation of an operator conducting an MRI scan with the relevant guidelines, such as procedure archetypes.
- WP6_AWI_HEA_REQ07 The human-machine interaction model should be able to handle actions in the physical world that are outside of the control of the system, and still adapt and give proper guidance to the operator.
- WP6_AWI_HEA_REQ11 The AdCoS should support dynamic sharing of model or situation identification between operator and system based on image representations or similar data.
- WP6_AWI_HEA_REQ25 The AdCoS should adapt to both medical and procedural context when the operator requests remote assistance.
- WP6_AWI_HEA_REQ35 After adding new features automatic tests should show that the entire system, including the UI, is still running as it is supposed to run.
- WP6_AWI_HEA_REQ43 After checking in with the user's credentials the UI can automatically adapt to their personalised settings.
- WP6_IGS_HEA_REQ03 The system shall be able to represent activities that are performed by operators. It includes estimated, execution times, periodicity, staff involved, prerrequisites…
- WP6_IGS_HEA_REQ09 The platform shall ease methods and tools to measure the usability of application
- WP6_IGS_HEA_REQ10 decision The platform shall ease the model and implementation of decision making
- WP6_PHI_HEA_REQ08 Tooling shall allows fast iteration to rapidly validate various concepts interactively

- WP7_HON_AER_REQ30 The system should provide a consistent and intuitive user interface, within and across the various hosted applications; including, but not be limited to, data entry methods, colour-coding philosophies, and symbology.
- WP7_HON_AER_REQ44 The system should be designed to minimise the occurrence and effects of flight crew error and maximise the identification and resolution of errors; for example, terms for specific types of data or the format in which latitude/longitude is entered should be the same across systems. Data entry methods, colour-coding philosophies, and symbology should be as consistent as possible across the various hosted EFB applications. These applications should also be compatible with other flight crew compartment systems.
- WP7_HON_AER_REQ45 The EFB system should be capable of alerting the flight crew of probable EFB system failures.
- WP7_HON_AER_REQ46 "The system should provide feedback to the user when user input is accepted. If the system is busy with internal tasks that preclude immediate processing of user input (e.g. calculations, self-test, or data refresh), the EFB should display a 'system busy' indicator (e.g. clock icon) to inform the user that the system is occupied and cannot process inputs immediately.

- The timeliness of system response to user input should be consistent with an application's intended function. The feedback and system response times should be predictable to avoid flight crew distractions and/or uncertainty."
- WP7_HON_AER_REQ49 The positioning and procedures associated with the use of the EFB should not result in unacceptable flight crew workload. Complex, multi-step data entry tasks should be avoided during take-off, landing, and other critical phases of the flight. An evaluation of the EFB intended functions should include a qualitative assessment of incremental pilot workload, as well as pilot system interfaces and their safety implications.
- WP7_HON_AER_REQ66 Create a common GUI that will allow to show dynamics logs, physiology recordings, event lists etc. at one time and that will allow for annotations of a situation.
- WP7_HON_AER_REQ71 Create a tool that is able to automatically evaluate a quality of an artifact according to general rules. The artifact may be defined as a screenshot or element description etc.
- WP7_HON_AER_REQ78 Create a tool/methodology that is able to classify an action of agent (human, machine) being either appropriate or erroneous. It is assumed that the tool has a task/procedure model with all supported alternate actions for a given situation.

- WP8_ADS_CTR_REQ17 The system shall be able to analyze the status and workload of adjacent HQs and subsequently offer support to transfer events to them
- WP8_ADS_CTR_REQ18 The system shall be able to analyze the workload of operators in one HQ and subsequently offer support to the supervisor to redistribute events among them
- WP8_ADS_CTR_REQ22 The system shall offer scaled functionality
- WP8_IRN_CR_REQ04 The AdCoS shall normalize the workload, either low or high, on the operator
- WP8_IRN_CR_REQ09 The AdCoS shall adapt to the competence and expertise level of the operator
- WP8_IRN_CR_REQ10 The AdCoS shall adapt to the psyco-physical status of the operator (e.g. high/low workload, time preassure,physical features)
- WP8_IRN_CR_REQ11 The AdCoS shall adapt with respect to the role assigned to each operator for incoming calls
- WP8_IRN_CR_REQ12 The AdCoS shall adapt to the language competences of the caller
- WP8_IRN_CR_REQ13 The AdCoS shall adapt to the geographical localization of the caller and of the target installation
- WP8_IRN_CR_REQ15 The AdCoS shall adapt to the frequency of incoming calls
- WP8_IRN_CR_REQ17 The AdCoS shall adapt to the priority level of the malfunctioning detected and the type of service addressed
- WP8_IRN_CR_REQ18 The AdCoS shall adapt to the asynchronous between the call and the malfunctioning detection
- WP8_IRN_CR_REQ19 The AdCoS shall adapt to the historical intervention gathered on a target installation
- WP8_IRN_CR_REQ20 The AdCoS shall adapt to the number of operators available

- WP9_TAK_AUT_REQ07 The urgency of a blind-spot warning should be determined and then communicated to the blind-spot audio/visual feedback interface
- WP9_TAK_AUT_REQ08 Blind-spot indicator designs shall reflect the actual situation (speeds and relative positions of objects) and propose appropriate actions to the user by indicating directions and user actions that avoid collision.
- WP9_TAK_AUT_REQ09 Blind-spot detection should be reliable and detection failures (failure to detect object in blind-spot/wrong detection of object in blind-spot) should be minimized. If complete reliance is impossible, prediction of the reliability by the driver should be supported by avoiding irregularity of detection failures. The reliance shall have low specifity and be applicable to all driving conditions.
- WP9_TAK_AUT_REQ18 Ideally, the system shall use a combination of critical events, operator performance measures, operator modeling and physiological assessment of the operator to determine timing of automation mode transitions.
- WP9_TAK_AUT_REQ25 The urgency of an ACC warning should be measured and then communicated to the ACC audio/visual/haptic feedback interface
- WP9_TAK_AUT_REQ27 ACC detection should be reliable and detection failures (failure to detect object in front/wrong detection of object in front) should be minimized. If complete reliance is impossible, prediction of the reliability by the driver should be supported by avoiding irregularity of detection failures. The reliance shall have low specifity and be applicable to all driving conditions.
- WP9_TAK_AUT_REQ35 If the user ignores the lane departure/blind spot warning and continues to steer off road/into traffic the system shall issue an alarm through an appropriate channel. It shall also propose counter-actions to relax the situation.
- WP9_TAK_AUT_REQ36 If the user senses the automation is decelerating the vehicle and uses the brake pedal, the ACC should be deactivated and longitudinal control re-issued to the driver.
- WP9_TAK_AUT_REQ37 If the user senses the automation is decelerating the vehicle and uses the accelerator pedal, the accelerator pedal shall provide a to be determined resistance and a warning shall be issued by the system.
- WP9_TAK_AUT_REQ64 System shall use minimal correction (just the amount necessary to avoid collisions) for user input error. User shall be supported in finding and avoiding false input by preventing input that will lead to undefined conditions, informing about corrections and giving the opportunity to postpone error treatment for non-critical errors.
- WP9_IFS_AUT_REQ02 The RTP platform should allow to support replaying of simulations cases/tests
- 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
- WP9_IFS_AUT_REQ04 Having a virtual car able to be dynamically piloted by the driver model
- WP9_IFS_AUT_REQ05 Having road environments and traffic events corresponding to the WP9 scenarios, where the driver model can drive a virtual car.
- WP9_IFS_AUT_REQ06 Driver mental model building / updating in a synchronized way with the Simulated Road Environment and Event (traffic scenarios)

- WP9_IFS_AUT_REQ10 Virtual simulation of car sensors (radar, camera, telemeter) as components of AdCos1 to be simulated and tested with the RTP during the Project
- WP9_IFS_AUT_REQ11 Target System definition (to be simulated) and algorithms to be developed for driver monitoring and adaptive & cooperative assistance / HMI
- WP9_IFS_AUT_REQ12 Virtual simulation of car sensors (radar, camera, telemeter) as component of AdCos2 to be simulated and tested with the RTP during the Project
- WP9_CRF_AUT_REQ09 "When the driver has indicated his/her intention to change lane and there is not a side lane, or there is a side obstacle, or there is an incoming obstacle from the rear on the side lane, the driver should be warned so that he/she does not start the lane change maneuver. Driver's state shall be considered as well."
- WP9_CRF_AUT_REQ16 When the driver is facing at the same time with more conditions that could generate an indication or a warning from the system, only the most critical indication should be given to the driver.
- WP9_OFF_AUT_REQ02 The Bayesian driver model must be able to update its initial (offline) learned parameters using inputs of the driver (steering angle, brake pedal position, throttle position) and available sensor data while driving assisted.
- WP9_OFF_AUT_REQ04 The Bayesian driver model must be able to return meaningful results after a fixed amount of computation time.
- WP9_OFF_AUT_REQ09 After an initial offline learning phase, the driver model must be able to classify the currently shown driving style (e.g. aggressive, sporty, ecp, normal) with aCorrect Classification Rate (CCR) of (80÷85)% and provides information about the driver's profile (e.g. mean speed, mean TTC).
- WP9_IAS_AUT_REQ06 The driver shall be able to override the automatic longitudinal control at any time. In case the driver applies the brake, the automated system shall turn off for the duration
- WP9_IAS_AUT_REQ10 The automatic action of the automated system shall not be interrupted in case the driver operates the steering wheel manually, but taken into account by the automated system.
- WP9_IAS_AUT_REQ11 The automated vehicle shall be able to change the lane for an overtaking maneuver. It shall adapt the speed according to the traffic in the neighboring lane and maintain a safe spacing to other traffic participants.

5 Conclusions

This document describes the objectives and the methodology of the process of requirement analysis regarding their relevance for model based analysis techniques and tools. The methodology has been applied on requirements collected from applications work packages (WP6 to WP9) and results are reported in Annex 1 file.

Annex 1: Requirements

Selected requirements are described in detail in annexed file "Holides-WP4-D4_1-v1.0.xlsx".

References

Baziuk W (1995). "BNR/NORTEL path to improve product quality, reliability and customer satisfaction" – In Proc. Of the 6th Int. Symp. On Soft. Reliability Eng

Boehm B. (1976) "Software Engineering" –– In IEEE Trans. On Comp and Soft. Eng.

Clarke, E. M., Kroening, D., Sharygina, N., & Yorav, K. (2005). SATABS: SAT-based predicate abstraction for ANSI-C. Vol. 3440. LNCS, Springer-Verlag.

Cousot, P., & Cousot, R. (1992). Abstract interpretation frameworks. Journal of Logic and Computation, 2(4), 511–547.

Edelstein, O., Farchi, E., Nir, Y., Ratsaby, G., & Ur, S. (2002). Multithreaded Java program test generation. IBM Systems Journal, 41, 111–125.

Fiedor, J., & Vojnar, T. (2012). ANaConDA: A framework for analysing multi-threaded C/C++ programs on the binary level. LNCS, Springer-Verlag.

Henzinger, T. A., Jhala, R., Majumdar, R., & Sutre, G. (2003). Software verification with blast. Vol. 2648. LNCS, Springer-Verlag.

IEC 62366 (2007). Medical devices - Application of usability engineering to medical devices.

HoliDes DOW "Description Of Work" – 2013

ISO/IEC/IEEE 15288 (2008). System engineering and software engineering – System life cycle processes.

Kam, J. B., & Ullman, J. D. (1976). Global data flow analysis and iterative algorithms. Journal of the ACM, 23, 158–171.

Nielson, F., & Nielson, H. R. (1999). Type and effect systems. Vol. 1710. LNCS, Springer-Verlag.

Visser, W., Havelund, K., Brat, G., Park, S., & Lerda, F. (2003). Model checking programs. Automated Software Engineering Journal, 10(2), 203–232.

*** End of document ***