Kelsch, J., Temme, G., Schindler, J.: (2013): *Arbitration based framework for design of holistic multimodal human-machine interaction.* Contributions to AAET 2013, 6.-7. Feb. 2013, Braunschweig, Germany, ISBN 978-3-937655-29-1

Arbitration based framework for design of holistic multimodal human-machine interaction

Johann Kelsch, Gerald Temme, Julian Schindler

German Aerospace Center / Institute of Transportation Systems

Lilienthalplatz 7, 30108 Braunschweig, johann.kelsch@dlr.de

Abstract

Creating dynamic cognitive systems, such as human-machine systems with the appropriate interaction inside, the system engineer is usually facing a high overall complexity. For instance, designing a modern driver assistance system, the developer is dealing with the diversity of drivers' behavior, the complexity of the traffic and vehicle dynamics and the heterogeneity of already designed assistance systems. To deal with this complexity effectively, the creation of holistic, stable and well-usable cognitive systems implies the usage of a design framework, which could provide benefit and orderliness to the creation process. In this contribution, we propose such a design framework that consists of the theoretically derived generic cognitive system architecture as well as of the arbitration-based methodology and tool-based interface for the interaction design. Besides the theoretical background, we describe the general rules of the framework in a stepwise manner using an automotive example of a newly developed driver assistance system for cooperative lane changes in a highway scenario.

Acknowledgements

We thank Matthias Heesen, Raphael Klemm (DLR) and Markus Zimmermann (TUM) for their fruitful participation in the development of the interaction design for the exemplary lane change assistance system.

Introduction

The main goal of this paper is to contribute to the structured design process in the research field of cognitive systems engineering (CSE) [16] by proposing a correspondent design framework. CSE is an important part of systems engineering [6], since the designed systems get more complex and

because the human is a crucial part of those systems. During the last three decades, much effort has been made addressing the CSE research field and bringing up many useful concepts. One of the achievements (and maybe the most important) was the definition of the machine as a cognitive agent [16], although there is criticism from the perspective of human sciences [22] regarding the inability of the machines to reason. Though, the today's machines are able to percept and to comprehend complex situations, for instance, by data fusion. Machines are able to project situation dynamics, for example, probabilistically, in order to generate autonomous decisions and control actions. In other words, today's machines can be called cognitive, because they possess already placeholders for all levels of situation awareness [7] as well as internal knowledge and behavioral models. However, in this contribution, we use the term cognitive system 'CS' as an aggregation for the cognitive multi-agent and human-machine systems. Further, we use the acronym 'CA' for all kinds of cognitive agents: humans, intelligent machines, automation, assistance etc. Furthermore, the important keywords and key-phrases will be highlighted by the *italic* font.

The next important achievement in the field of CSE was the user centered design approach toward the usability engineering [26], [27], which moved the consideration of the human user's cognition further into the CS design process and balanced it versus technology centered design [13]. After investigating several design metaphors for highly automated human-machine systems in the automotive domain [12], [17], it became also more obvious that the machine design is not forced to be human-like but human-compatible [11], [19]. Further, the methodology toward cognitive function and task analysis [3] and, in more general, cognitive work analysis [23], [41], [45] has to be considered. These approaches allow analyzing CAs and CS in a holistic way considering the possible cognitive processes and states within the CAs. During the proposed *functional CS decomposition*, the functional CA and CS requirements can be gathered and clarified.

Though, besides the functional requirements in the CS design, there is also a strong necessity of gathering the *structural* requirements and thus the necessity of *structural CS decomposition*. It is not enough to answer the questions, for example, when and which tasks should be allocated to which CA and which goals which CA is supposed to achieve by means of which resources? To move toward the more concrete system design and implementation it is also crucial to answer the following two research questions. *1. How to implement the CS structurally?* That means in detail: How can theoretically known cognitive structure elements be mapped to the soft- and hardware of the developer? Which structure elements have to be

mapped? What is the appropriate common data representation thereby? There are many theoretical considerations regarding CA architectures [4], [30], [32] and several of them regarding CS architectures [31], [39], but unfortunately, we discovered a lack of structurally applicable propositions on the CS level as well as on the level of integrated CA interaction design process. Supposedly, this circumstance is due to the lack of the evident *structural similarities* between different kinds of CS both on the system and on the agent interaction level. What are cognitive structural elements that could be common for the agent interaction at an office work place, in a driver-automation system and in a swarm of UAV's? By means of those similarity hints, essential structural information could be gathered to describe a generic CS architecture. Further, a proper design interface to the generic CS architecture could be derived regarding creation or modification the functional CS inner life by means of the CA interaction design. That leads us directly to the second research question. *2. How can the interaction design process be connected to a generic CS architecture?* We will deal with both research questions in two theoretical parts of this contribution.

After the theoretical parts, we will focus on how to translate the proposed theory toward a more concrete CS architecture and interaction design for the automotive domain. We will give a brief and stepwise overview about the application of the proposed framework for the development of new cooperative lane change driver assistance system. Besides this, we will emphasize the advantages for a system engineer, who follows the proposed framework philosophy using a new interaction design interface tool based on the presented theoretical basis as well.

Structural similarities toward the generic CS architecture

Let's address the question about structural similarities between different CS first. Therefore, we take a look on the definitions: What are a CS and a CA and what is the dependency between them? The most obvious statement, that can be made after defining the machine as a CA, is the conclusion that every CS, either artificial or not, consists of at least two CAs. Surely, generic CS with only one CA inside is possible as well, but in these brief theoretical considerations we do not take into account this exceptional case. However, there are two CAs within a generic human-machine system: a human and a machine agent and there are at least a driver and an assistance system agent within a generic driver-automation system in automotive domain. This initial CS description is well-known and is also identified in the field of the multi-agent system research, such as distributed artificial intelligence [15], [30], [32], [39], which, by the way, can be used in CS

design by means of the provided powerful methodology for the agent modeling. Further, it is important to note that a CS with several CAs inside is a *complex system* because of the theoretically assumed emergent origin of the cognition itself [5], [22]. So in CSE, we have to deal with emergent effects, highly non-linear dynamics, self-organizing processes etc.

There are several definitions of the term 'cognitive agent', for instance [5] "agent is any entity *able to act*", [32], [40], etc. Most definitions claim nearly the same sub-parts of a CA: beliefs, desires, goals, plans, intentions and *actions* and a 'cognitive kind' of dealing with these items by a CA. For instance, a CA can possess declarative and procedural knowledge, mental models, a CA can be able to learn something, a CA can transform allocated tasks into *activity* by means of cognitive functions [3] and a CA can possess internal situation awareness states with a generic cognitive informational processing inside: perception, comprehension, projection [7] and (more or less) independent decision and *action* performance. Thereby, the cognitive items, such as beliefs, desires, goals etc., can be regarded as factors influencing the cognitive process itself as well as products of and informational links between the processing steps. Though, on the current abstract level of CS consideration, these cognitive items can be *aggregated as actions*. Further, to deal with the similar content on the CS level, it is reasonable to define the environment within a particular CS as a non-cognitive (reactive) agent (NCA) like proposed also in [5]. In automotive domain, we have at least two NCAs: the environment and the vehicle. Thus, the *first structural similarity* toward a generic CS architecture is that *all (re-)acting entities within a CS can be represented by (re-)acting agents*.

However, in this contribution we will not stress the whole agent modeling side of the CSE. Rather, we will stress that first: *agents are interfering with each other by means of actions*, second: that *a system engineer can influence this interference* and third: that a generic CS architecture and corresponding interaction design interface can be derived using that information. Because of this and in order to give the system engineer the freedom to integrate her/his own agents or agent models into the particular CS design, we use CAs as well as NCAs structurally as black boxes with corresponding inputs and outputs. Such a 'black box' is able to perceive information from other agents, to process this information changing certain internal states, making decisions and providing the processed information toward interference with other agents. An example for the automotive domain: During the driving, a driver is acting and interacting constantly with her/his assistance, vehicle and environment and the assistance agent does so as well. The vehicle is reacting and interacting with the reactive

environment and with the CAs on board. In such a CS model the *second structural similarity* is that the *informational behavior of agents can be described by agents' input and output (re-)actions.* In this manner, all agents can be regarded being in an *action space* interfering with each other by means of actions. We speak about *interaction* between two agents, when input and output (re-)actions of each agent have both-sided connections.

Every behavior of an agent contains an informational component. "One cannot not communicate" [43]. And the information itself, provided by an agent, needs an initial *event* to be presented and/or to be revealed. Thereby, *events can be regarded as trigger for new actions as well as connections between agents and actions. Actions themselves can be regarded as expression of internal agent states and can produce new events.* This relationship between event, (cognitive) agent state and action is the *third structural similarity* on the way to the generic CS description. Until now, we (over-)simplified the CS ontology toward the abstract interaction model [36] and similar to the 'quasi thermodynamic' model of Brownian Agents [37]. Figure 1 left shows a cooperative lane change traffic scenario and on the right the corresponding structural CS decomposition.

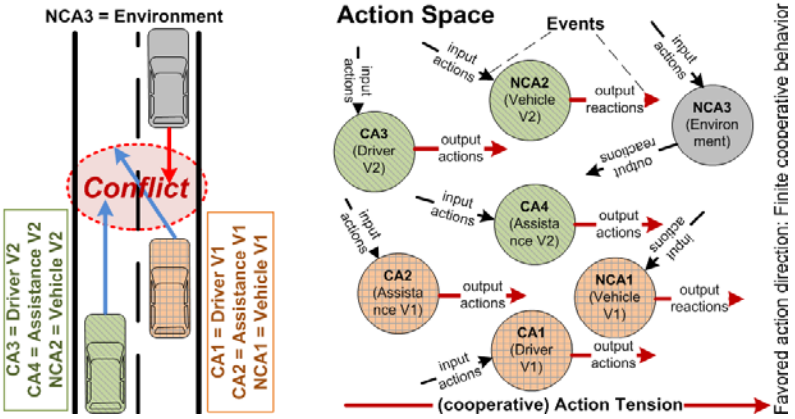

Figure 1: Conflict modeling in a cooperative lane change scenario (right) and (left) structural decomposition of the scenario into the agent and action space based model

There are two main vehicles: V1 (NCA1) on the right lane and V2 (NCA2) on the left lane of a two-lane highway. Both vehicles have human drivers and lane change assistance systems on board of V1 and V2 (CA1-4). The third vehicle and the environment are aggregated in the non-cognitive agent NCA3. In this scenario, a *conflict* is modeled: V1 has to change its current right lane because of the dangerous environmental issue in front, but the

aimed left lane is potentially blocked by the fast moving vehicle V2. To resolve this conflict situation, there is a necessity for *cooperation* between V1 and V2. On the Figure 1 right, the agents are shown as circles, their actions as arrows and events as arrowheads. However, to get more concrete toward the structural CS and agent interaction description, we need five further theoretical concepts: *action tension* [19], *agent interference* [5], *arbitration* [18], *multi-resource theory* in cognitive informational processing [44] and the *time horizon* in CA behavior.

Let's speak about the action tension first. There are two well-accepted psychological concepts (Psychological Forces [21] and Affordances [13]) dealing with human's motivational processes. They operate with terms being near to physical and technical terms, such as force and tension. Both concepts can be summarized in a prospective and directed metric *action tension* indicating the direction toward certain possible actions to reach the same system state (Figure 2 left). This provides the opportunity to model the behavior of CAs and CS both on a higher level and on a *common ground*. The complex interaction processes, being highly dependent on CA behavior (actions) itself, can be modeled in a more simplified manner.

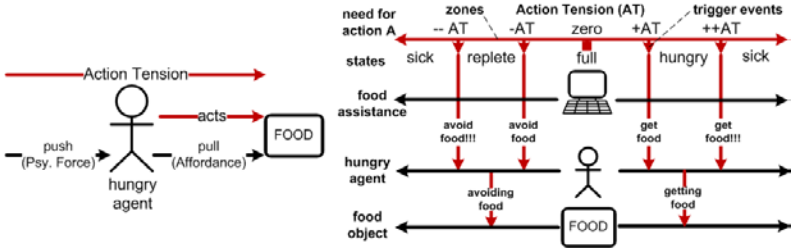

Figure 2: Combination of psychological force and affordance to action tension (left) and action tension based hypothetical example of a food assistance (right)

A hypothetical example of using action tensions in CS design is shown as 'food assistance' in a sequence diagram on Figure 2 right. The machine agent communicates relevant interaction signals to the hungry agent in order to keep her/him in the optimal state 'full' and away from the state 'sick'. Here is also shown that the action tension can have *two opposite directions*: toward get/avoid food (+/++, -/-- AT), or steer left/right as example for automotive applications. So, the *continuous two-directional value separated in discrete zones would be the common data representation* in the proposed CS architecture. Here, the value itself is the action tension, the zones are operationalized tension states and the borders between the zones mark the trigger events for possible cognitive state changes and anchor points for artificially designed interaction and CS behavior synchronization signals.

Thesis: A cognitive system being in a particular state can be described as controlled by multiple tensions directed toward actions leading to other system states. Hypotheses: I) Humans are supposed to act toward keeping the tensions in an optimal state and machines can be designed to do so as well [19]. II) The particular action tension falls if the CA acts toward the optimal action tension state and it rises when a) CA acts contrary to the optimal action tension state or b) CA persists (performs no action) being in a suboptimal state. Hence, the particular action tension can describe in the generalized way how urgent the particular action is and which sequence of interactions is necessary to reach the change of the system state toward minimizing the action tension by transition into the aimed optimal action tension state. *The consideration of action tensions is the fourth structural similarity*. It describes the 'missing glue' between all possible CA actions within the action space and could be also one of the applicable "pre-cognitive structures" claimed to be formalized for CSE [5]. Further, action tensions can be used in CSE *to analyze* and *to redirect* the CA actions by means of explicitly designed interaction signals in the direction favored by the system engineer. This is indicated on Figure 1 right by parallel drawn actions influenced by the correspondent (cooperative) action tension.

Using action tensions as universal reference values for the CS design and control, the system complexity can be reduced and the control processes can be linearized toward favored design and control issues, such as workload distribution, attention control, situation awareness [8], safety, efficiency, comfort improvement etc. Therefore, action tensions can be separated in time, space or more complex value dependent zones operationalized, for example, by inductive exploration and/or by theoretical deduction of possible agent states and subsequently connected agent (re-)actions. The borders between the zones can be used as events synchronizing the CA actions by artificially designed signals, which can be distributed within the whole CS, for example, by machine agents or *HMI agents* [2]. Thereby, the important constraint of using the action tensions for the CS and interaction design is the finding of an appropriate operationalization manner.

The fifth structural similarity can be found in the *CA interference*, which is necessary for deriving the generic CS architecture as well as for the corresponding interaction design interface. Here, we follow Castelfranchi [5] arguing that *all agents (CAs as well as NCAs) are interfering with each other within a CS by means of actions.* This way, the *behavior of a CS can be defined by the CA interference*. There are two dual sorts of interference: *positive* and *negative*. The negative interference causes emergent effects within a CS influencing negatively the objective CS performance during the

CA interference. The negative CA interference is characteristic for (potential) conflict states. For example, when a driver drives down a road fork by an intelligent vehicle and the driver and the machine agent on board have or are going to have different intentions. The driver could aim at steering to the left and the machine agent to the right. Will the vehicle move to the left or to the right? Those conflicts or competition situations and hence the negative interference between CAs can be managed by a concept of *arbitration* [18], which is a subset of the overall CA interactions. The particular arbitration model for the automotive domain is shown on Figure 3 left. The CA arbitration is defined as the finite *negotiation* between two CAs by means of proper interaction strategies and signals aimed at reaching a *joint intent, decision and action* within the CS and within the *available time*. This way, a system engineer can influence strategically and/or immediately the (joint) action of at least two CAs by explicitly designed arbitration strategies and signals. Therefore, the complete agents' actions must be described within a *common frame of reference* (CFoR) in order to manage the CA interference explicitly, for example, by an *arbiter* [18] or by another kind of HMI agent. It is important to note, that the CA interference is present at all steps of the agent informational processing. If we assume, that CAs possess an internal situation awareness processes, then the agents are interfering in perception, comprehension, projection, decision and action states using the corresponding cognitive items: beliefs, desires, goals, plans, tasks, intentions and actions. The (potential) conflicts on all levels of situation awareness can be arbitrated by influencing these cognitive items using action tensions. Therefore, a system engineer can moderate the joint CA decisions toward the joint CA actions by using additionally injected interaction signals, which must be defined within the CFoR as well and which can be triggered by events placed along the action tensions.

The positive interference is dual to the negative one. However, it can be managed exactly the same way like the negative interference. The positive interference causes emergent effects within a CS, which influence positively the objective CS performance during the CA interference. An example for the positive interference is the agent cooperation [5], [14]. To improve, for example, the overall effectiveness of the traffic in the proposed scenario from Figure 1 left, it is reasonable to initiate a cooperation process between V1 and V2. So, the V2 could decelerate slowly and thus energy efficiently in order to allow the lane change for V1. Afterwards, V1 could change the lane fluently and efficiently as well. Therefore, during the functional CS decomposition, a priori domain purpose and functions [23] must be defined, such as "the traffic should act energy efficiently" and/or "the dangerousness of the traffic situations should be minimized". If those functions or purposes

are in a possible conflict state, for example, the traffic is not yet acting efficiently, then the system engineer can use this discovered higher level conflict as the starting point to improve the CS performance. This can be done, for example, by means of cooperation mechanisms, such as cooperation modes [14] or design patterns for cooperation [1].

To initiate, moderate and decide within cooperation processes, arbitration concept can be used, because finite negotiations of cooperative (joint) decisions and actions within cooperation processes are necessary as well. For the cooperation, and thus for the positive interference issues, HMI agents, such as arbiters, can be used. Further, cooperation processes and cooperative CA actions can be managed by the system engineer on all levels of cognitive informational processing by using operationalized (cooperative) action tension within the cooperation design. Following this line of argumentation, we state that within the proposed CS ontology, *it is possible to address every kind of CA interference by arbitration*.

We claim that the *starting point for design of CA interference management and thus the starting point for interaction design should be the analysis and modeling of (possible) conflicts together with the functional and structural CS decomposition*. Thereby, the system engineer could follow the proved minimalistic design principles, such as "keep it simple but not simpler" or "never touch a running system", because only possibly discovered CS conflicts would justify the usage of new interaction strategies and signals. *If no (possible) system conflicts are discovered, then no CA interaction design is necessary!* Further, the discovered conflicts allow using design patterns as proved 'problem-solution dyads' [1]. Furthermore, such conflict-centered design approach allows a balancing between human-centered and technology-centered CS design, since *the relevant usability problems within a CS arise within the action space between the CAs* [28].

The *sixth structural similarity* can be found considering the *time horizon in CA behavior* within a CS. Rasmussen [33] argues, for example, that the human's control behavior is separable into three levels: skill-, rule- and knowledge-based levels. For the interaction design, he proposes using signals, signs and symbols respectively. The main argumentation thereby is that for the processing of those interaction signals on the proposed levels of control more processing time or longer *time constants* are necessary, when the informational processing moves from the signal to the symbol processing and from the skill-based to the knowledge-based level. In the theory of situation awareness [7], there are three levels as well: perception, comprehension and projection that are assumed as a sequence of the cognitive informational processing. Because of necessity of running through

former situation awareness levels (for instance, perception and comprehension), the next level of situation awareness (for instance, projection) would take an overall longer time slot and it could be controlled using longer time constants as well. Other considerations due to the time horizon in CA behavior we find in [5] and based on it [14], which propose at the end three levels as well: action-, plan- and meta-level. These levels can be regarded also with rising time constants on the way from the action- to the meta-level. Similar triples regarding the time horizon, we find in economic science and military definitions: operational, tactical and strategic levels, which have also rising time constants on the way from operational to the strategic level, e.g., of planning or executing tasks. However, because in system and interaction design the system engineer is interested in effects s/he can achieve by the proper interaction strategies and parameterization of the interaction elements, it is reasonable to cluster all discussed terms after the time horizon similarity. After doing this, we define also three levels regarding the expected influence to the agent (re-)actions: long-term, short-term and immediate effect levels (Figure 3 right). Exemplary for automotive CS, on the long-term level could be the management of automation levels [11], cooperation modes [14] and/or maneuver arbitration. On the short-term level could be trajectory arbitrations (lateral deviation, speed and distance to other objects etc.) On the immediate level, the management of control allocations to particular CAs could be placed.

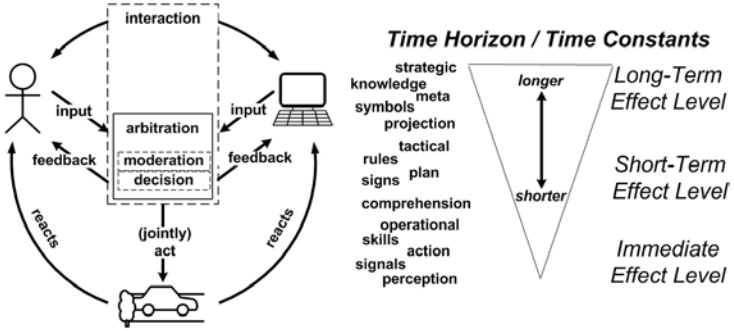

Figure 3: Particular arbitration model for automotive domain (left), time horizon/time constants dependent clustering of cognitive processing models (right)

The last *seventh structural similarity* toward the generic CS architecture can be found in the parallel consideration of the multi-resource theory (MRT) [44] and channel structures proposed in technical communication sciences [38]. MRT argues that humans possess several different cognitive resources, which can be used simultaneously. When humans percept and process information toward actions subconsciously, symbolically, linguistically by

means of different modalities, such as acoustic, visual or haptic, then we are able to process the information in parallel with some interference or noise effects between the particular cognitive resources. For interaction design, MRT shows the opportunity to use the parallel cognitive processing power by channeling information separating the designed interaction signals into former defined multimodal channels and sub-channels.

Every multimodal channel, e.g. haptic channel, can be separated in sub-channels dealing with different sorts of interaction signals and correspondent semantics. So, a 'hint sub-channel' can be defined providing hints by discrete haptic signals, such as haptic ticks, or an 'action guiding sub-channel' can be defined providing information by continuous haptic force feedback. Other modalities, such as visual and acoustic, can be separated in sub-channels with deterministic semantics as well. This way by means of sub-channels, different semantics can be channeled to the CAs simultaneously using the power of CA parallel cognitive informational processing. Further, the deterministic kind of providing information semantics allows using technical approaches for managing the informational flows, such as prioritization, de-conflicting and/or arbitration. Furthermore, early consideration of semantics brings orderliness, consistency and user compatibility to the design process as well as to the interaction design itself.

Now, it is possible to set up the generic CS architecture (Figure 4 left). On the upper (Agent Action and Tension) layer we deal with the first four CS structural similarities. Here, the inference of agent states and actions as well as inference of possible events and action tensions can be placed. This can be done using former definition of CFoR and concepts described above. Brief summary: All entities in a CS are agents with internal (cognitive) states. The agent states are getting 'visible' by correspondent actions. All agent actions and thus all agent states can be manipulated on later layers using synchronization events and correspondent interaction signals placed along operationalized action tensions. On the next (Agent Interference Management) layer we deal with the fifth and sixth CS structural similarities. Here, agent interference management units, such as arbiters or other sorts of HMI-Agents, can be placed arbitrating discovered system conflicts and/or managing the agent interference toward cooperation. This can be done using the appropriate interference management strategies on the three levels dependent on the expected time horizon of action influence effects: long-term, short-term or immediate and derived from former designed informational syntax and semantics.

On the next (Interaction Element and Channel) layer, very concrete implementation of interference management can be placed in form of concrete interaction element (and hence multimodal sub-channel) behavior definition, parameterization, calculation, de-confliction and channeling toward the last (Agent Interaction Hardware) layer. Thereby, the interaction hardware implies the human user compatible active multimodal inceptor/feedback devices as well as the hardware providing interaction to and between the machine agents. Further, the Semantic Design Layer can be connected to the proposed generic CS architecture by means of the tool-based Interference and Element Design Interface. We will deal with this important interface as well as with the correspondent interaction design methodology in the next section.

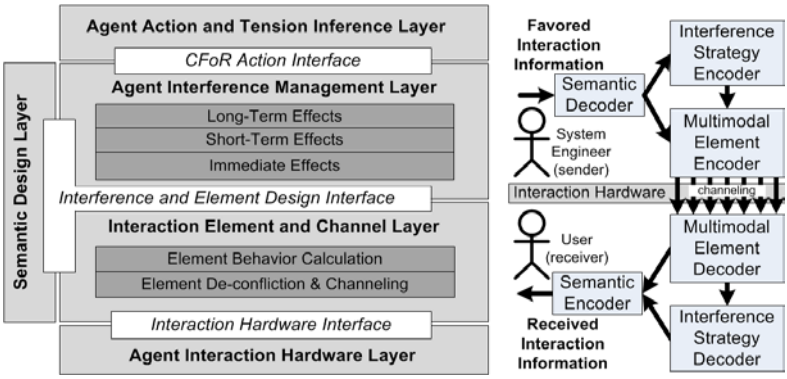

Figure 4: Generic CS architecture with connected interaction design interface (left) and communication path between the system engineer and the system user (right)

Interaction design process and interface to the CS architecture

On the Figure 4 right a communication path is shown, which describes the informational flow between a system engineer designing favored interaction and a human user or other kind of CA receiving the designed information. First, the system engineer has to decode or to translate the favored information on the sematic and syntax design layer answering the central question: *"Which information must be communicated to the CAs?"* The answer to this question can be found modeling situation conflicts and using deductive and/or inductive methods. Deductive methods could be the cognitive task, function and/or work analysis [3], [34], [41]. Inductive methods could be the systematic design space definition and/or exploration. However, the answer to the first central question implies the answers to the sub-questions: "Are there issues that the system engineer can resolve by a proper interaction design? What will happen if these issues stay unresolved?

Which issues must be resolved necessarily? When and which information CAs need therefore? What could be the applicable arbitration strategies?" Under the assumption that it is possible to linearize and to operationalize the agent interference process by using action tensions, the output of the semantic decoder part should be the clearly decomposed and consistent semantic informational pieces, such as phrases and sentences, preferable in form of a 'semantic storyboard' and/or a sequence diagram dealing with informational and agent behavioral flows on the semantic level.

Afterwards, the storyboard can be used toward the particular CA interaction design answering the second central question: "*How the identified information design can be communicated to the CAs?"* Therefore, the favored interaction design should be decomposed structurally answering the following sub-questions: "Which action tensions can be identified and operationalized within the gathered semantics and the particular CS context? Which CA actions must be exhibited (initiated, assisted, gained etc.) or inhibited (prevented, forbidden, redirected etc.)? Which interference and/or arbitration strategies and signals can be used to resolve the discovered issues? Which multimodal channels and sub-channels can be used therefore?" Thereby, the storyboard can be encoded in action tension dependent strategies using multimodal elements, which should be separated into sub-channels and tuned to be presented to the user by means of the interaction hardware. The aimed CA receives the channeled interaction information by decoding the multimodal element signals and strategies encoding them into the overall designed semantic meaning at the end.

Such kind of holistic interaction design often needs the participation of and the discussion between persons working in various disciplines: Designers, technicians, human factors professionals or computational scientists. In order to make the design process easily understandable and usable for all involved disciplines, it is necessary to present and to design the interaction in a clearly arranged way. Figure 5 left shows a proposition for a design interface that can be easily connected between the Agent Interference Management and the Interaction Element and Channel Layer of the former proposed generic CS architecture (Figure 4 left). After producing the semantic storyboard and definition of action tensions it is possible to describe the interference strategies (syntax) using ternary logic [20], [24] between the trigger events based on action tensions and the triggered interaction elements separated in correspondent semantic channels. The used ternary logic consists of three logical states: enable (+), disable (-) and don't care (0). The truth tables of the logic are shown on the Figure 5 right.

The activation of a particular semantic sub-channel and consequently the enabling of a particular multimodal interaction element can be logically determined by the ternary AND-relation. The element priority based de-confliction of the elements placed on the same semantic channel can be determined by the ternary OR-relation. Elements themselves can be parameterized by manipulating correspondent element parameters. This way, using the proposed design interface, a deterministic interaction model can be produced, which will be integrated into the generic CS architecture. This model can be tested and tuned within the particular developed CS and within the particular used simulation environment. Usage of the proposed design interface and principles, e.g. in a GUI, could provide an easy and open parameterization and tuning interface for the interaction design, a quick portability of the interaction models between different CS environments, a reusability of interaction design models, a well-ordered dialog between participating professionals from different disciplines and so guide the whole design process toward holistic cognitive system design.

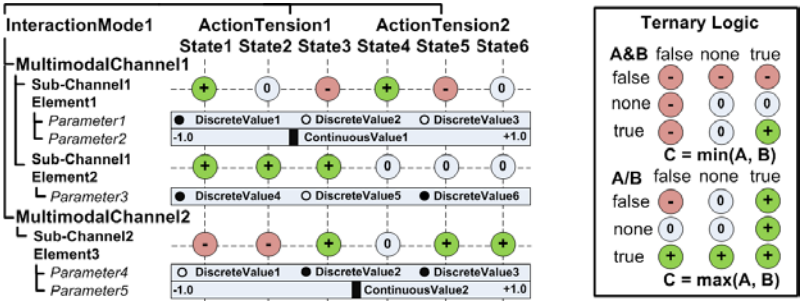

Figure 5: Tool-based interaction design interface to the generic CS architecture (left) and truth tables for AND- and OR-relations in used ternary logic (right)

Application of the interaction design framework

In this section, we deal with the exemplary application of the proposed interaction design framework. Therefore, we use the cooperative lane change scenario introduced on Figure 1 left. The initial step toward the development of new cooperative lane change assistance was the problem analysis and the conflict modeling. Therefore (1), we defined [23], [41] for the current domain (land-based cooperative traffic) the domain purpose (personal and goods transport on highways) and the domain values & priorities (safety, efficiency and comfort). Then (2), we modeled the former mentioned scenario that is very common for the domain on the one hand but conflicts the domain values on the other hand. The modeled CS scenario is unsafe, inefficient and uncomfortable. The solution for the modeled conflict

was expected in the cooperation between the vehicles V1 and V2 initiated by the assistance systems on board. Therefore in the aimed CS, all possible CA actions must be influenced toward cooperative actions by an appropriate interaction design. Following the proposed framework (3), we assumed a linear cooperative action tension that controls cooperative CA actions within the aimed CS. That tension must consist of a particular state sequence describing the syntax of the cooperation process as well as of the state transitions triggering events for additionally injected signals. Those signals were supposed to provide the aimed interaction semantics for the cooperation maintenance and to synchronize the cooperative CS behavior. Further, the optimal tension was defined using two functional system states: 'There is no need for cooperation' and 'cooperation is done' (Figure 6 mid).

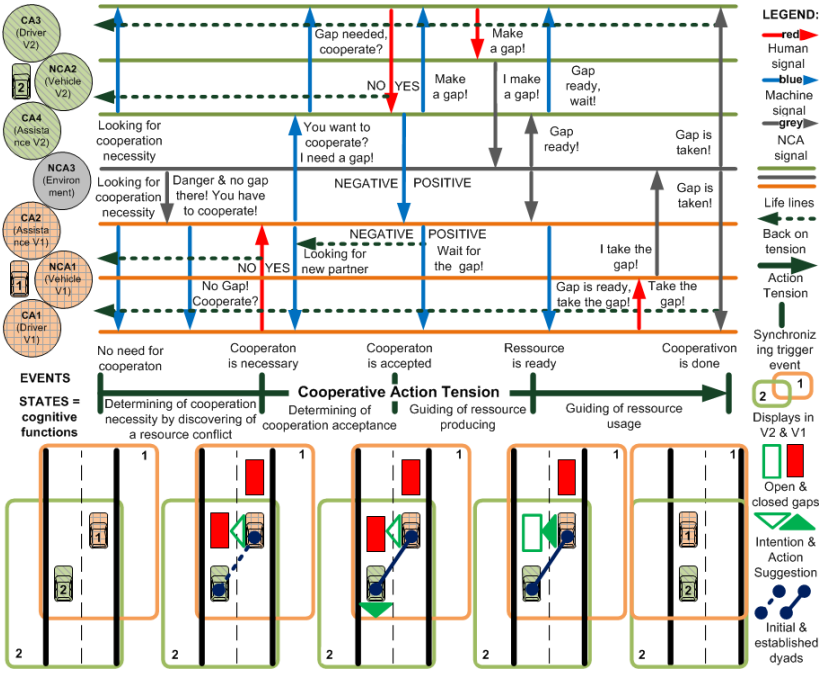

Figure 6: Framework based design for cooperative lane change assistance system: Semantic storyboard and agent interaction signal flow (top), after cognitive function and task analysis operationalized cooperative action tension (mid), design of the visual feedback display for both vehicles (down) and overall legend (right)

The next development steps were: (4) functional and structural decomposition of the origin CS, (5) interaction design and (6) composition of the aimed CS regarding the operationalized cooperative action tension

and the designed interaction. The steps (4), (5) and (6) we will describe in more detail. Thereby, we will focus on four important design aspects: *initiation of the cooperative process*, *arbitration by using the cooperation modes*, by the *providing distributed situation awareness* and by the *cooperative action guiding*. Figure 1 right shows the structural decomposition, where all CAs as well as NCAs were identified. Using the functional decomposition of the origin CS and working out the semantic storyboard for aimed interaction design (Figure 6 top) in parallel, it was possible to identify the functional states and synchronizing events in between, which operationalize the assumed cooperative action tension (Figure 6 mid). Four main states (or action tension zones) were identified: 'Determining of cooperation necessity', 'determining of cooperation acceptance', 'guiding of resource producing' and 'guiding of resource usage'. The events between the states were identified as: 'Cooperation is necessary', 'cooperation is accepted' and 'resource is ready'.

During the functional CS decomposition, in particular during the cognitive task analysis toward the aimed system composition, two cooperation modes [14] for implementation of the aimed interaction design were defined: '*Perception mode*' and 'mutual control: *Action suggestion mode*' (Figure 7 bottom right). In perception mode the driver is responsible for the execution of the complete lane change maneuver. If necessary, the assistance is responsible for initiation of the cooperation. If cooperation usage is accepted by the drivers, the cooperation mode changes to the action suggestion mode. Here, the assistances negotiate opening the gap and suggesting to the drivers the proper actions, which lead to the successful completion of cooperation toward opening the gap and changing the lane.

Technically, this behavior is implemented as two HMI-Agents called *Mode Selection and Arbitration Unit (MSAU)* and *Maneuver Arbitration Unit (MAU)*. MSAU manages the cooperation mode selection and interference, which is a strategic and long-term sort of CS control. The driver can decide strategically to continue driving, e.g. performing of the next maneuver, with or without support by the assistance. Internally, the MSAU is a state machine (Figure 7 top right), which describes the available cooperation modes and all possible transitions between these modes. Primarily, it uses the design pattern of interlocked transitions [35] and the operationalized cooperative action tension. It activates HMI elements in order to arbitrate the mode selection conflicts and to inform the driver about the current cooperation mode and mode transition. In action suggestion mode, the MSAU communicates the current cooperation mode to the MAU that activates maneuver guiding HMI elements using cooperative action tension.

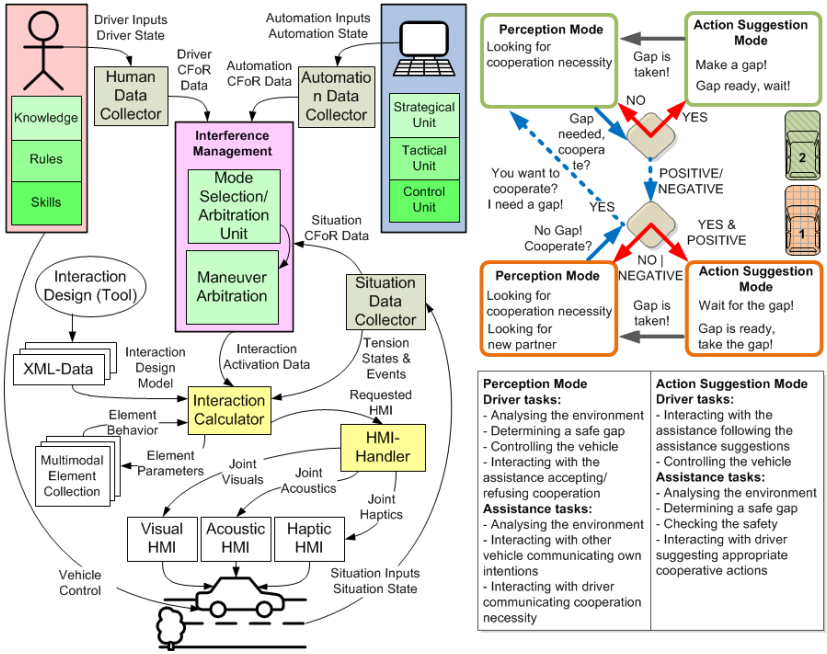

Figure 7: Adaptation of the generic CS architecture for cooperative lane change assistance system (left). Particular behavior of the Mode Selection/Arbitration Units in vehicles V1, V2 and definition of the used cooperation modes (right)

Composing the aimed CS structurally, both HMI-Agents (MSAU&MAU) were implemented within an integrated, reusable and model-based C++ software/hardware architecture (Figure 7 left) called *InteractionController*. It follows the generic CS architecture from Figure 4 left, automotive arbitration model from Figure 3 left and is integrated into the DOMINION implementation platform [25]. For better reusability of software modules, the *InteractionController* was implemented using software modeling tool BOUML 2.7.6 [29]. Human, automation and situation data collectors together with the CAs (driver and automation) represent the Agent Action and Tension Inference Layer of the generic CS architecture. MSAU&MAU are producing long-term effects on the Agent Interference Management Layer. They provide interaction activation data for the Interaction Element and Channel Layer, where Interaction Calculator and HMI-Handler are placed. Interaction Calculator uses the *interaction design model* and the object oriented multimodal element collection to produce the requested HMI signal flow separated in the former defined (sub-) channels. The HMI-

Handler arbitrates this flow and produces a deterministic output for the Agent Interaction Hardware Layer represented by the HMI-hardware.

Designing the CA interaction, the main strategy was to initiate the cooperation process between the vehicles V1 and V2, to maintain it and to guide it along the cooperative action tension toward the successful termination. This was mainly done using the design pattern and the visual channel of the distributed situation awareness. Figure 6 down sketches the corresponding visual display in both vehicles with slightly different parts of the same situation display. The own vehicle is always in the middle. Further, it shows the interaction sequence (syntax) depending on the cooperative action tension. Therefore, the defined semantic sub-channels were: Cooperation initiation and existence feedback (dashed and solid dyads), intention and action suggestion feedback (hollow and full triangles) and gap availability feedback (hollow and full rectangles). Using dyads was supposed to prevent the diffusion of responsibility [42] and to maintain the cooperation process. Intention and gap availability feedback was supposed to improve the distributed situation awareness. Action suggestion feedback was supposed to initiate and to guide the cooperative actions. According to the concept presented on Figure 5, the model of the presented interaction was developed in a GUI-based interaction design tool (Figure 8). The tool itself was developed in a straightforward way using the FLTK graphical toolkit [10]. This design tool is arranged in different layers grouping and logically linking different types of relevant interaction data. The data shown in the GUI is represented internally in eXtensible Markup Language (XML) [9] as a human readable hierarchical format. This makes it possible to reuse interaction design models and use them without or with other GUIs as well.

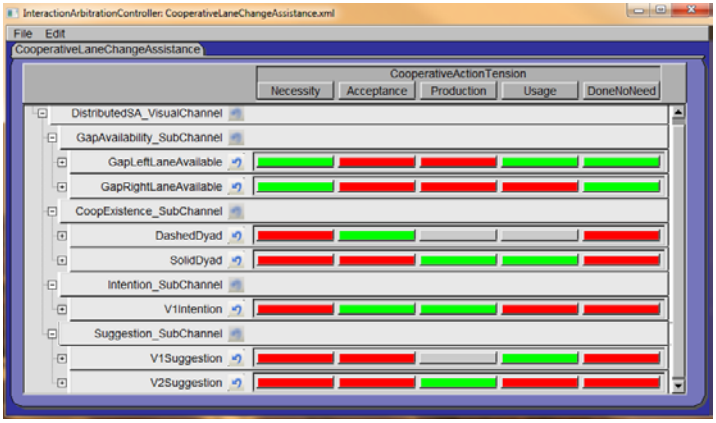

Figure 8: GUI-based Interaction design tool with exemplary interaction implementation

Summary and Outlook

In this contribution, we presented the theoretical background to the arbitration based framework for design of holistic multimodal human-machine interaction. Based on seven CS structural similarities and ternary logic, it consists of a vertical generic CS architecture and a horizontal tool-supported interaction design interface and methodology. Further, we showed an exemplary application of the proposed framework toward development of a cooperative lane change assistance system for the automotive domain. The work on the presented framework as well as on the cooperative lane change assistance is still in progress. It is planned to formalize the framework toward verifiable CA interaction modeling and the correspondent usability testing methodology. The presented exemplary interaction design and the developed cooperative assistance system will be verified next in a usability study and a usability experiment as well.

References

[1] Baumann, M., Cao, Y., Cauchard, F., Corradini, P., Dehais, F., Fonda, S., Heers, R., Heesen, M., Kelsch, J., Losi, G., Magnaudet, M., Montanari, R., Müller, H., Neujahr, H., Rister, F., Tango, F, Temmos, J.-M., Tesauri, F., Zimmermann, M. (2012): *Reference Designs and Design Patterns for Cooperation & DCoS State Inference and Adaptation.* D3-03 public deliverable for EU-D3CoS

[2] Boy, G. (1994): *Interface Agents for handling fuzzy descriptors in information retrieval.* 5th Int. Conference on Processing and Management of Uncertainty in Knowledge-Based Systems, Paris, France, July 4-8, 1998

[3] Boy, G. (1998): *Cognitive Function Analysis for Human-Centered Automation of Safety-Critical Systems.* Conference on Human Factors in Computing Systems, Los Angeles, California, USA, Apr. 18-23, pp. 265-272

[4] Byrne, M. D., (2001). *ACT-R/PM and menu selection: Applying a cognitive architecture to HCI.* International Journal of Human-Computer Studies, 55, pp. 41-84

[5] Castelfranchi, C. (1998). *Modeling Social Action for AI Agents.* Journal Artificial Intelligence - Special issue: artificial intelligence 40 years later archive Volume 103 Issue, Aug. 1-2, 1998

[6] Elm, W. C., Gualtieri, J. W., McKenna, B. P., Tittle, J. S., Peffer, J. E., Szymczak, S. S., Grossman, J. B. (2008): *Integrating Cognitive Systems Engineering Throughout the Systems Engineering Process*. Journal of cognitive Engineering and Decision Making, Vol. 2, Num. 3, Fall 2008, pp. 249-273

[7] Endsley, M. R. (1995): *Toward a theory of situation awareness in dynamic systems.* Human Factors 37, pp. 32-64

[8] Endsley, M. R. (2003): *Designing for Situation Awareness: An Approach to User-Centered Design*. Taylor & Francis

[9] Extensible Markup Language (XML) 1.0 (Fifth Edition): http://www.w3.org/TR/REC-xml, last visited Jan. 2013
[10] Fast Light Toolkit (FLTK): http://www.fltk.org, last visited Dec. 2012
[11] Flemisch F., Kelsch J., Löper C., Schieben A., Schindler J. (2008): *Automation spectrum, inner/outer compatibility and other potentially useful human factors concepts for assistance and automation*. de Wart, D. (Ed). Human Factors for Assistance and Automation, Shaker Publishing
[12] Flemisch, F. (2000): *The horse metaphor as a guideline for vehicle automation*. Proposal to the National Research Council, Munich
[13] Gibson, J. J. (1977): *The Theory of Affordances. In Perceiving, Acting, and Knowing*, Shaw, R. & Bransford, J. (Eds.), ISBN 0470990147
[14] Hoc, J. M. (2001): *Towards a cognitive approach to human-machine cooperation in dynamic situations.* International Journal of Human-Computer Studies, 54, pp. 509-540
[15] Hollan, J., Hutchins, E., Kirsh, D. (2000): *Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research*. ACM Transactions on Computer-Human Interaction, Vol. 7, No. 2, pp. 174-196
[16] Hollnagel, E. & Woods, D. D. (1982): *Cognitive Systems Engineering: New wine in new bottles.* International Journal of Human-Computer Studies, Volume 51, Number 2, Aug. 1999, pp. 339-356
[17] Holzmann, F. (2007): *Adaptive Cooperation between Driver and Assistant System.* Springer
[18] Kelsch, J. (2012): *Arbitration between Driver and Automation: why Overriding is just the Tip of the Iceberg*. InteractIVe Summer School, 04.-06.06.12, Corfu Island, Greece
[19] Kelsch, J., Heesen M., Hesse T., Baumann M. (2012): *Using human-compatible reference values in design of cooperative dynamic human-machine systems.* EAM 2012, 11-12.09.2012, Braunschweig, Germany
[20] Kleene, S. C. (1938): *On notation for ordinal numbers.* Journal of Symbolic Logic, Vol. 3, No. 4, pp. 150-155
[21] Lewin, K. (1938): *The Conceptual Representation and the Measurement of Psychological Forces.* Psychological Theory, 4, Duke University Press, Durham, N.C.
[22] Lintern, G. (2007): *What is a Cognitive System?* Proceedings of the 14th International Symposium on Aviation Psychology, pp. 398-402. 18.-21.04.2005, Dayton
[23] Lintern, G. (2011): *Tutorial: Work Domain Analysis*. www.cognitivesystemsdesign.net
[24] Łukasiewicz, J., Borkowski, L. (Ed.) (1970): *Selected Works*, Warsaw
[25] Montenegro, S., Dannemann, F., Dittrich, L., Vogel, B., Noyer, U., Gacnik, J., Hannibal, M., Richter, A., Köster, F. (2010): *(Spacecraft Bus Controller + Automotive ECU) / 2 = Ultimate Controller*, in Lecture Notes in Informatics (GI-Edition), 160, Gesellschaft für Informatik. Software Engineering 2010 / ENVISION2020, pp. 103-114, Paderborn
[26] Nielsen, J. (1993): *Usability Engineering.* Morgan Kaufmann, San Francisco
[27] Norman, D. A. & Draper, S. (1986): *User centered system design: New perspectives on human-computer interaction.* Mahwah, Lawrence Erlbaum Associates
[28] Norman, D. A. (2007): *The design of future things.* New York, N.Y. Basic Books
[29] Pagès, B. (2012): *Bouml 2.7.6.* http://bouml.free.fr/, last visited Dec. 2012
[30] Puerta, A., Eisenstein, J. (2002): *XIML: a common representation for interaction data.* Proceedings of the 7th international conference on Intelligent user interfaces, 214-215
[31] Putzer, H. & Onken, R. (2003): *COSA - A generic cognitive system architecture based on a cognitive model of human behavior*. CTW (2003) 5: pp. 140-151
[32] Rao, A. S. & Georgeff, M. P. (1991): *Modeling rational agents within a BDI-architecture.* Proceedings of the International Conference on Knowledge, Representation and Reasoning, pp. 473-484
[33] Rasmussen, J. (1986): *Information processing and human-machine interaction: An approach to cognitive engineering.* New York, North-Holland, pp. 101-115

[34] Rasmussen, J., Pejtersen, A., Goodstein, L. (1994): *Cognitive Systems Engineering*. New York, Wiley
[35] Schieben, A., Temme, G., Köster, F., Flemisch, F. (2011): *How to interact with a highly automated vehicle - generic interaction design schemes and test results of a usability assessment.* Human Centred Automation. de Waard, D., Gerard, N., Onnasch, L., Wiczorek, R. & Manzey, D. (Eds.). Maastricht, Shaker Publishing: pp. 251-266
[36] Schramm, W. (1954): *How Communication Works.* In: W. Schramm (ed.). The Process and Effects of Mass Communication. Urbana: University of Illinois Press 1955, 3-26
[37] Schweitzer, F. (2003): *Brownian Agents and Active Particles.* Springer, Berlin
[38] Shannon, C.E. (1948): *A Mathematical Theory of Communication*. Bell System Technical Journal, Vol. 27
[39] Shoham, Y., Leyton-Brown, K. (2009): *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations.* Cambridge University Press
[40] Vakarelov, O. (2009): *The Cognitive Agent.* Department of Philosophy, University of Arizona, Tucson, Arizona
[41] Vicente, K. J. (1999): *Cognitive work analysis: Toward safe, productive, and healthy computer based work*. Lawrence Erlbaum Assoc Inc
[42] Wallach, M. A., Kogan, N., & Bem, D. J. (1964). *Diffusion of responsibility and level of risk taking in groups.* Journal of Abnormal and Social Psychology, 68, pp. 263-274
[43] Watzlawick, P., Helmick Beavin, J., Jackson, D.D. (1967): *Pragmatics of Human Communication: A Study of Interactional Patterns, Pathologies, and Paradoxes*. Norton
[44] Wickens, C.D. (1984): *Processing resources in attention*. In Parasuraman R. & Davies D.R. (Eds.), Varieties of attention, pp. 63-102. New York. Academic Press
[45] Woods, D. D., & Hollnagel, E. (1987): *Mapping cognitive demands in complex problem-solving worlds*. International Journal of Man-Machine Studies, 26, pp. 257-275