AASE - Actor-Actor Systems Engineering
Theory & Applications
Micro-Edition (Vers.1)

eJournal: uffmm.org, ISSN 2567-6458
8.Jan 2018
Email: info@uffmm.org

Gerd Doeben-Henisch*
and
Zeynep Tuncer†
and
Louwrence Erasmus‡
and
Khalid Idrissi

8.Jan 2018

Contents

1 Different Views 3

2 Philosophy of the AAI-Expert 3

3 Problem (Document) 4

4 Check for Analysis 5

5 AAI-Analysis 5
- 5.1 Actor Story (AS) 5
  - 5.1.1 Textual Actor Story (TAS) 5
  - 5.1.2 Pictorial Actor Story (PAT) 6
  - 5.1.3 Mathematical Actor Story (MAS) 6
  - 5.1.4 Simulated Actor Story (SAS) 7
  - 5.1.5 Task Induced Actor Requirements (TAR) 7
  - 5.1.6 Actor Induced Actor Requirements (UAR) 8
  - 5.1.7 Interface-Requirements and Interface-Design 8
- 5.2 Actor Model (AM) 8
  - 5.2.1 Design Principles; Interface Design 9
- 5.3 Simulation of Actor Models (AMs) within an Actor Story (AS) 9
- 5.4 Assistive Actor-Demonstrator 9
- 5.5 Approaching an Optimum Result 9

6 What Comes Next: The Real System 10
- 6.1 Logical Design, Implementation, Validation 10
- 6.2 Conceptual Gap In Systems Engineering? 10

7 The AASE-Paradigm 11

Abstract

Based on the the ISSN-Publication *AAI - Actor-Actor Interaction. A Philosophy of Science View* from 3.Oct.2017 and version 11 of the ISSN-Publication *AAI - Actor-Actor Interaction. An Example Template* this paper transforms these views in the new paradigm *Actor-Actor Systems Engineering* understood as a theory as well as a paradigm for and infinite set of applications. In analogy to the slogan 'Object-Oriented Software Engineering (OO SWE)' one can understand the new acronym *AASE* as a systems engineering approach where the actor-actor interactions are the base concepts for the whole engineering process. Furthermore it is a clear intention to view the topic AASE explicitly from the point of view of a *theory* (as understood in Philosophy of Science) as well as from the point of view of possible *applications* (as understood in systems engineering). Thus the classical term of *Human-Machine Interaction (HMI) or even the older Human-Computer Interaction (HCI)* is now embedded within the new AASE approach. The same holds for the fuzzy

*Frankfurt University of Applied Sciences (FRA-UAS) and Institut for New Media (Frankfurt)(INM)
†Technical University Darmstadt (TUD)
‡Council for Scientific and Industrial Research (CSIR) of South-Africa

disciplin of *Artificial Intelligence (AI)* or the subset of AI called *Machine Learning (ML)*. Although the AASE-approach is completely in its beginning one can see how powerful this new conceptual framework already is. [1]

[1] This text has a long 'conceptual history' leading back to the Philosophy-of-Science studies of Doeben-Henisch 1983 - 1989 in Munich under the guidance of Peter Hinst, many intensive discussions between Doeben-Henisch and Erasmus about Systems engineering since 1999, a paper written by Doeben-Henisch and Wagner 2007 [DHW07] with ongoing discussions since then, a lecture by Doeben-Henisch about formal specification and verification in 2010 [DH10], two papers by Erasmus and Doeben Henisch in 2011 [EDH11b], [EDH11a], about 20 regular semesters with the topic Human-Machine Interaction by Doeben-Henisch at the Frankfurt University of Applied Sciences (Frankfurt, Germany)(unpublished) in the timespan 2005 - 2015, two regular semesters with the topic HMI together with Tuncer in SS2016 and WS2016 at the Frankfurt University of Applied Sciences (Frankfurt, Germany) (unpublished), and two workshops with Erasmus in summer 2016 and Spring 2017 (unpublished). Additionally many discussions between Doeben-Henisch and Idrissi about AI and HMI since 2015.

1 Different Views

If one wants to deal with the development of optimal interfaces within certain tasks for executing actors[2] one can distinguish different *views* onto this problem.

The common *work view* in systems engineering is an *expert (EXP)* as part of a *systems engineering process (SEP)* who takes a *problem description* D_p and does some analysis work to find an *optimal solution candidate (OSC)*.

One *level above* we have the *manager (MNG)* of the systems engineering process, who is setting the *framework* for the process and has to *monitor* its working.

Another upper level is the *philosopher of science (POS)* who is looking onto the managers, processes, and their environments and who delivers *theoretical models* to *describe* these processes, to *simulate* and to *evaluate* these.

In this text the *Actor-Actor Interaction (AAI)* is the main focus, embedded in a *Systems Engineering Process (SEP)*, all embedded in a minimal *Philosophy of Science (PoS)* point of view.

For this the following minimal SEP-structure is assumed:

$$\begin{aligned}
SEP(x) &\quad iff \quad x = \langle P, S, Sep \rangle & (1)\\
Sep &\quad : \quad P \longrightarrow S\\
Sep &\quad = \quad \alpha \otimes \delta \otimes \mu \otimes \upsilon \otimes o\\
\alpha &\quad := \quad Analysis\ of\ the\ problem\ P\\
\delta &\quad := \quad Logical\ design\\
\sigma &\quad := \quad Implementation\ of\ S\\
\upsilon &\quad := \quad Validation\\
o &\quad := \quad Deployment
\end{aligned}$$

The outcome of the analysis of an AAI-expert is an *optimal solution candidate (OSC)* for an interface of an assisting actor embedded in a complete *behavior model* M_{SR} given as an *actor story (AS)* combined with possible *actor models (AMs)*. This output provides all informations needed for a following *logical design*. The logical design provides the blue-print for a possible *implementation* of a concrete working system whose behavior should be in agreement (checked through a *validation* phase) with the behavior model provided by the AAI-analysis.

[2] Today still mostly human persons.

2 Philosophy of the AAI-Expert

Before digging into the details of the following actor-actor interaction (AAI) analysis done by an AAI-expert one has to consider the conditions under which the AAI-expert is doing his job.

1. The executing AAI-experts are *human actors* or *machine actors*. If machine actors then it is assumed, that they posses at least *human-level intelligence*.

2. A *viewpoint* is a part of of an everyday situation where a *observer* is located in some *three-dimensional space* and is *looking* onto a section of the space with his visual system. This results in some *visual perception* $Per_{vis,obs}$ of the situation in the observer.

3. An observer is embedded in a *time-line* $TIME$ which represents a fourth dimension. Every visual perception $Per_{vis,obs}$ of the situation in an observer can be aligned with an interval of the time-line represented as a *technological time* T_{tech} realized by some *clock* $CLCK$.

4. The visual perception $Per_{vis,obs}$ of an observer is structured by distinguishable *properties* Π embedded in *spatial relations* where the observer is a *point of reference* ('left/ right', 'before/ behind', 'above/ below', ...). Subsets of the properties can be *grouped* as *objects* OBJ. Objects inherit the spatial relations ('object A is above object B', 'object A is left from object B', ...). Objects

can *observational be embedded* in other objects, inducing some *visual structure* into an object (a 'house-object' has embedded a 'window-object' and the window-object can have further embedded objects). Embedded objects can also be part of spatial relations ('the window-object left from the door-object below the roof-object').

5. An observer A can *communicate* his *own perceptions* to another observer B only if both observers use a language L_{obs} *common to both*. This requires that an expression $e \in L_{obs}$ which encodes a visual property of observer A $p_{vis}(A)$ does the same for observer B. This means $p_{vis}(A)$ should be the same as $p_{vis}(B)$. That this is possible requires the following mappings: $meaning : Per_{vis,A} \longleftrightarrow L_{obs}$ and $meaning : Per_{vis,B} \longleftrightarrow L_{obs}$. Because the visual perceptions *inside* an observer are no direct objects, the communicating observers can *judge* the *similarity* of their intended visual perceptions only if these *inner* (= *subjective*) perceptions are somehow *causally related* to some phenomena which are *external* (= *objective*) to all observers. If this is the case, then the participating observers can check the similarity of their subjective perceptions by recurring to the causally related external objects. If observer A correlates his subjective perception $Per_{vis,A}$ with expression e and this expression in turn with an external object o and the other observer B connects his subjective perception $Per_{vis,B}$ with expression e and this expression in turn with the external object o, then both can assume that they *mean the same*, otherwise the situation is undefined.[3] This fact that the subjective visual perception *does not count* as long as it cannot be connected to a *commonly shared external object* has favored the practice to define the *meaning of an expression* e only by recurrence to the external (objective) matters without mentioning the subjective perceptions. For restricted practical applications this can work, but for a philosophical analysis this is not enough.

6. For a language of observation L_{obs} one needs therefore expressions for *properties*, *objects* as sets of properties, and *spatial relations* between the objects. This mapping of perceptions triggered by a real situation into formal expressions creates a *set of expressions* which represent only a *subset* of all the possible properties of the real situation. To make this difference explicit the subset of property-expressions is called a *state*. To represent *changes* happening in the real situation in the formal representation one relates states on a time-line one after the other. The real changes are thereby encoded by deletions of properties or by the creation of new properties between two succeeding states.[4]

3 Problem (Document)

1. The *problem document* D_P is the result of a communication between some *stakeholder (SH)* and some *experts*, which have discussed a *problem* P which the stakeholder wants to be solved.

2. Additionally to the general problem a finite set of special *constraints (C)* can be given.

3. Due to the fuzziness of human communication one has to assume a certain degree of a *semantic gap* with regard to the participants of this communication as well as for potential readers of the document.

[3] Although this inference from the agreement in external matters to the similarity of the corresponding inner states of the participants is very common, it is no proof! It is a working hypothesis which works quite well in many situations. But it is conceivable that the corresponding inner states of the participants can be completely different in their realizations; as long as they *function equivalently* the participants will not be able to detect this difference.

[4] For a more detailed formal account of these considerations see se page `https://uffmm.org/2018/01/08/actor-actor-systems-engineering-aase-formal-appendix/`

4 Check for Analysis

Within the general analysis phase of systems engineering the AAI-perspective constitutes a special view. This implies a check of the *occurrence* of the following aspects:

1. At least one *task (T)* and
2. an *environment (ENV)* for the task and
3. an *executive actor (ExecA)* as the intended user.

5 AAI-Analysis

The goal of the AAI-analysis is to find an optimal *assistive actor (AssA)* to support the *executive Actor (ExecA)* in his task. For this to achieve one needs an iterative application of the whole AAI-analysis process whose results are evaluated for an *optimal solution*.

To analyze the problem P one has to dig into the problem P so far that one is able to *tell a complete story*, how to *understand* and later to *realize* the task.

It can be some work to *investigate* the details of such a story. The investigation is complete if the resulting story is *sound*, that means all participants agree that they *understand* the story and that they *accept* it.

To *communicate a story* we assume the following main modes: *textual*, *pictorial*, *mathematical*, as well as *simulated*.[5]
.

5.1 Actor Story (AS)

To *communicate* a story in the main modes textual, pictorial, mathematical as well as simulated one has to consider the above mentioned epistemological situation of the AAI-expert.

The *point of view* underlying the description of an actor story AS is the so-called 3^{rd}-person view. This means that all participating objects and actors are described from their *outside*. If an actor *acts* and changes some property through it's action it is not possible in a 3^{rd}-person view to describe the inner states and inner processes, that enabled the actor to act and why he acts in this way. To overcome the limits of a 3^{rd}-person view one has to construct additional models called *Actor Models (AMs)*. For more details have a look to the section 5.2.

5.1.1 Textual Actor Story (TAS)

An actor story AS in the *textual mode* is a *text* composed by expressions of some *everyday language* L_0 – default here is English L_{EN} –. This text describes as his *content* a sequence of distinguishable *states*. Each state s – but not an *end-state* – is connected to at least one other follow-up state s' caused by the *change* of at least one *property* p which in the follow up state s' either is *deleted* or has been *newly created*.

Every described state s is a set of properties which can be sub-distinguished as *objects (OBJ)* which are occurring in some *environment (ENV)*. A special kind of objects are *actors (As)*. Actors are assumed to be able to *sense* properties of other actors as well as of the environment. Actors are also assumed to be able to *respond* to the environment without or with taking into account what *happened before*.

Actors are further sub-divided into *executive* actors as well as *assistive* actors. Assistive actors A_{assist} are those who are expected to support the executive actors A_{exec} in fulfilling some *task (t)* (with $t \in T$).

A *task* is assumed to be a sequence of states with a *start* state s_{start} and a *goal* state s_{goal}, where the goal-state is an end state. The set of states connecting the start and the goal state is finite and constitutes a *path* $p \in P$. There can be more than one path leading from the start state to

[5] For an extended explanation of the formalisms used in this document see the web-page `https://uffmm.org/2017/12/27/formal-appendix-for-the-aai-case-studies/`

the goal state. The states between the start and the goal state are called *intermediate* states.

Every finished actor story has a least one path.[6]

5.1.2 Pictorial Actor Story (PAT)

In case of an textual actor story (TAS) – as before explained – one has a set of expressions of some common language L_0. These expressions *encode a possible meaning* which is rooted in the *inner states (IS)* of the participating experts. Only the communicating experts *know* which meaning is encoded by the expressions.

This situation – labeled as semantic gap – can cause lots of misunderstandings and thereby *errors* and *faults*.

To minimize such kinds of misunderstandings it is a possible strategy to *map* these intended meanings in a *pictorial language* L_{pict} which has sufficient resemblances with the intended meaning. Replacing the textual mode by a story written with a pictorial language L_{pict} can *show* parts of the encoded meaning more directly.

As one can read in the section 2 'Philosophy of the View-Point' the world of objects for a *standard user* is mapped into a spatial structure filled with properties, objects, actors and changes. This structure gives a blue-print for the *structure of the possible meaning* in an observer looking to the world with a 3^{rd}-person view. Therefore a pictorial language can *substitute* the intended meaning to some degree if the pictorial language provides real pictures which are structurally sufficient similar to the perceived visual structure of the observer.

To construct a *pictorial actor story (PAS)* one needs therefore a mapping of the 'content' of the textual actor story into an n-dimensional space embedded in a time line. Every time-depended space is filled with objects. The objects show relations within the space and to each other. Objects in space, the space itself, and the changes in time are based on distinguishable properties. To conserve a consistency between the textual and the pictorial mode one needs a *mapping* between these both languages: $\pi : L_0 \longleftrightarrow L_{pict}$.

5.1.3 Mathematical Actor Story (MAS)

To translate a story with *spatial structures* and *timely changes* into a mathematical structure one can use a *mathematical graph* γ extended with *properties* Π and *changes* Ξ for encoding.

A *situation* or *state* $q \in Q$ given as a spatial structure corresponds in a graph γ to a *vertex* v, and a *change* $\xi \in \Xi$ corresponds to a pair of vertices (v, v') which is directly connected by an *edge* $e \in E$.

If one maps every vertex $v \in V$ into a set of property-expressions $\pi \in 2^{L_\Pi}$ with $\lambda : V \longmapsto 2^{L_\Pi}$ and every edge $e \in E$ into a set of change-expressions L_Ξ with $\epsilon : E \longmapsto 2^{L_\Xi}$ then a vertex in the graph γ with the associated property-expressions can represent a state with all its properties and an edge e followed by another vertex v' labeled with a change-expression can represent a change from one state to its follow-up state.

A graph γ extended with properties and changes is called an *extended graph* γ^+.

Thus we have the extended graph γ^+ given as:

$$\gamma^+(g) \quad iff \quad g = \langle V, E, L_\Pi, L_\chi, \lambda, \epsilon \rangle \tag{2}$$
$$E \subseteq V \times L_\chi \times V \tag{3}$$
$$\lambda : V \longrightarrow 2^{L_\Pi} \tag{4}$$
$$\epsilon : E \longrightarrow 2^{L_\Xi} \tag{5}$$

Every assumed *object* $o \in OBJ$ attached to a vertex represents a sub-set of the associated properties. An *actor* $a \in A$ is a special kind of object by $A \subseteq OBJ$.

Some more remarks to a *change-event*:

[6] To turn a textual actor story into an *audio* actor story (AAS) one can feed the text into a *speech-synthesis* program which delivers spoken text as output.

The occurrence of a change is represented by two vertices v, v' connected by an edge e as $e : \{v\} \longmapsto \{v'\}$. The follow-up vertex v' has at least one property-expression less as the vertex v or at least one property-expression more. This change will be represented in a formal *change-expression* $\epsilon \in L_{\chi}$ containing a list of properties to be *deleted* as $d : \{p_1, p_3, ...\}$ and properties to be newly *created* as $c : \{p_2, p_4, ...\}$.

The *deletion-operation* is shorthand for a mapping of subtracting property-expressions like $d : \{s\} \longmapsto s - \{p_1, p_3, ...\}$ and the *creation-operation* is shorthand for a mapping of adding property-expressions like $c : \{s\} \longmapsto s \cup \{p_2, p_4, ...\}$. Both operations are processed in a certain order: first deletion and then addition, $change = d \otimes c$.

To keep the consistency between a textual and a pictorial actor story one needs a mapping from the pictorial actor story into the mathematical actor story and vice versa, $m_{p.m} : L_{pict} \longleftrightarrow L_{math}$.

.

5.1.4 Simulated Actor Story (SAS)

A *simulated actor story (SAS)* corresponds to a given *extended graph* γ^{+} by mapping the extended graph into an *extended automaton* α^{+}.

The usual definition of a finite automaton is as follows: $\langle Q, I, F, \Sigma, \Delta \rangle$ with

1. Q as a finite set of *states*
2. I $\subseteq$ Q as the set of *initial states*
3. F $\subseteq$ Q as the set of *final states*
4. Σ as a finite input *alphabet*
5. $\Delta \subseteq$ Q $\times$ Σ^{*} $\times$ Q as the set of *transitions*

If one *replaces/ substitutes* the *states* by *vertices*, the *input expressions* by *change-expressions* and the *transitions* by *edges* then one gets: $\langle V, I, F, L_{\chi}, E \rangle$ with

1. V as a finite set of *states*
2. I $\subseteq$ V as the set of *initial states*
3. F $\subseteq$ V as the set of *final states*
4. L_{χ} as a finite set of *input expressions*
5. $E \subseteq V \times L_{\chi} \times V$ as the set of *transitions*

Finally one extends the structure of the automaton by the set of property-expressions L_{Π} as follows: $\langle V, I, F, L_{\chi}, L_{\Pi}, E, \lambda \rangle$ with $\lambda : V \longrightarrow 2^{L_{\Pi}}$.

With this definition one has an extended automaton α^{+} as an automaton who being in state v *recognizes* a change-expression $\epsilon \in L_{\chi}$ and generates as follow-up state v' that state, which is constructed out of state v by the encoded deletions and/ or creations of properties given as property-expressions from L_{Π}. All state-transitions of the automaton α^{+} from a start-state to a goal-state are called a *run* ρ of the automaton. The set of all possible runs of the automaton is called the *execution graph* γ_{exec} of the automaton α^{+} or $\gamma_{exec}(\alpha^{+})$.

Thus the *simulation* of an actor story corresponds to a certain run ρ of that automaton α^{+} which can be generated out of a mathematical actor story by simple replacement of the variables in the graph γ^{+}.

5.1.5 Task Induced Actor Requirements (TAR)

Working out an actor story in the before mentioned different modes gives an outline of *when* and *what* participating actors *should do* in order to *realize* a *planned task*.

But there is a difference in saying *what* an actor *should do* and in stating *which kinds of properties* an actor *needs* to be able to show this required behavior. The set of required properties of an actor is called here the *required profile* of the actor A $RProf_{A}$. Because the required profile is depending from the required task, the required profile is not a fixed value.

In the general case there are at least two different kinds of actors: (i) the *executing* actor A_{exec} and (ii) the *assistive* actor A_{assis}. In this text we limit the analysis to the case where executing actors are *humans* and assistive actors *machines*.

5.1.6 Actor Induced Actor Requirements (UAR)

Because the required profile $RProf_{requ}$ of an executive actor realizing a task described in an actor story can be of a great variety one has always to examine whether the *available executing actor* A_{exec} with its *available profile* $RProf_{avail}$ is either in a *sufficient agreement* with the required profile or not, $\sigma : RProf_{requ} \times RProf_{avail} \longmapsto [0, 1]$.

If there is a *significant dis-similarity* between the required and the available profile then one has to *improve* the available executive actor to approach the required profile in a finite amount of time $\chi : A_{avail,exec} \times RProf_{requ} \longmapsto A_{requ,exec}$. If such an improvement is not possible then the planned task cannot be realized with the available executing actors.

5.1.7 Interface-Requirements and Interface-Design

If the available executing actors have an available profile which is in sufficient agreement with the required profile then one has to *analyze the interaction* between the executing and the assistive actor in more detail.

Logically the assistive actor shall assist the executing actor in realizing the required task as good as possible.

From this follows that the executing actor has to be able to *perceive* all necessary properties in a given situation, has to *process* these perceptions, and has to *react* appropriately.

If one calls the sum of all possible perceptions and reactions the *interface of the executing actor* $Intf_{A,exec}$ and similarly the sum of all possible perceptions and reactions of the assistive actor the *interface of the assistive actor* $Intf_{A,assis}$,then the interface of the assistive actor should be optimized with regard to the executing actor.

To be able to know more clearly how the interface of the assistive actor $Intf_{assis}$ should look like that the executive actor can optimally perceive and react to the assistive interface one has to have sufficient knowledge about how the executive actor *internally processes* its perceptions and computes its reactions. This knowledge is not provided by the actor story but calls for an additional model called *actor model*.

5.2 Actor Model (AM)

While one can describe in an actor story (AS) possible changes seen from a 3^{rd}-person view one can not describe *why* such changes happen. To overcome these limits one has to construct additional models which describe the internal states of an actor which can explain why a certain behavior occurs.

To enable such a *transparent interaction* between actor and environment it will be assumed that an actor is generally an *input-output system (IOSYS)*, that means that an actor has *inputs (I)* allowing some kind of *perceptions* of his environment as well as *outputs (O)* allowing changes, modifications in the environment. The sum of all inputs and outputs defines the *interface* of an input-output system, written $Intf(x)\ iff\ x = \langle I, O \rangle$. Furthermore it is assumed that every actor has some *behavior function* ϕ which determines how the actor will respond with an output given some inputs. More formally this can be written as follows:

Def: Input-Output System (IOSYS)

$$\begin{aligned} IOSYS(x) &\quad iff \quad x = \langle I, O, IS, \phi \rangle \\ I &\quad := \quad Input \\ O &\quad := \quad Output \\ IS &\quad := \quad Internal\ states \\ \phi &\quad : \quad I \times IS \longmapsto IS \times O \end{aligned} \tag{6}$$

and with explicitly mentioning the *interface*:

Def: Input-Output System (IOSYS)

$$\begin{aligned} IOSYS(x) &\quad iff \quad x = \langle I, O, INTF, IS, \phi \rangle \\ I &\quad := \quad Input \\ O &\quad := \quad Output \\ INTF(x) &\quad iff \quad x = \langle I, O \rangle \\ IS &\quad := \quad Internal\ states \\ \phi &\quad : \quad I \times IS \longmapsto IS \times O \end{aligned} \tag{7}$$

Thus the behavior function ϕ generates an output O depending from the actual input I and some internal states IS, and – this is reflexive – the behavior can again change the internal states IS such, that these are in another shape for a next response. This means that the same input can be followed by different responses depending from the internal states. This includes properties which often are called *learning* and *intelligence*.

Because the *inner states (IS)* of every real system are *not* directly observable it follows that all assumptions about possible inner states as well as about the details of the behavior function ϕ represent nothing else as a *hypothesis* which is given in the format of a *formal model*. The formal space for such hypothetical models is *infinite*.

The only *constraints* for some kind of *plausibility/ soundness* of such formal hypothetical models is given by the actor story which is defining a framework within which the hypothetic model has to be embedded.[7]

5.2.1 Design Principles; Interface Design

Given the actor model AM of an executive actor A_{exec} one can derive some *actor-based principles* $Ax_{A,exec}$, how the interface $Intf_{assis,B}$ of an intended assistive actor B should look like to enable an optimal performance with the executive actor A. To make the actor-based principles $Ax_{A,exec}$ as empirically sound as possible one needs sufficient empirical research of real actors doing jobs like those required in the actor story.

From the dependency of the executive-actor-based principles for the design of an assistive-actor interface it follows that the principles can only be as good as the presupposed model.

[7]The modern tool of *Neuroscience* can measure many real properties of real neurons, whose activity is assumed to underly the observable behavior. But the limits of these measurements combined with the still unknown complexity of the mapping between neural activity and observable behavior are not allowing today a completely defined empirical mapping. This weakness is even more amplified by the fact, that the factor of the *consciousness* filtering a small subset of practical helpful phenomena out from the complexity of the body is today also not yet sufficiently understood.

5.3 Simulation of Actor Models (AMs) within an Actor Story (AS)

Programming a real computer with actor models and an actor story allows the simulation of actor models embedded in an actor story.

5.4 Assistive Actor-Demonstrator

Given the design of the interface of an assistive actor one can realize a *demonstrator* based on such a design called $Demo(Intf_{assis,B})$. Every created demonstrator is a possible *candidate* for the optimal solution. To check it's 'value' one uses the demonstrator within an usability tests.

5.5 Approaching an Optimum Result

To approach a possible optimum for a finite set of demonstrators one applies a set of usability measurements – called 'usability test' – in an iterative process. A *usability test* UT realizes a mapping of given *demonstrators* D into a set of *usability values* V as follows $\upsilon_{UT} : D \longmapsto D \times V$. A usability test includes a finite set of objective as well as subjective sub-tests. The values V of one usability test are usually given as a finite set of points in an n-dimensional space V^n. Thus after a usability test υ_{UT} has been applied to a demonstrator one has an ordered pair (D, V).

To find the *relative best* demonstrator in a finite set of candidate demonstrators $\{(D_1, V_1), (D_2, V_2), ..., (D_m, V_m)\}$ one has to define a *measure* $\mu : 2^{V^n} \longmapsto V^n$ for the assumed finite many n-dimensional values $\{V_1^n, V_2^n, ..., V_m^n\}$

to compare these values and identify for this set an optimal value. Thus $\mu(V_1^n, V_2^n, ..., V_m^n)$ computes a certain $V_i^n \in \{V_1^n, V_2^n, ..., V_m^n\}$.

Applying this measure to the set $\{(D_1, V_1), (D_2, V_2), ..., (D_m, V_m)\}$ gives the best demonstrator of this set.

6 What Comes Next: The Real System

After the completion of the AAI-analysis after n-many iterations[8] one has an actor story AS in four modes {TAS, PAS, MAS, SAS}. Furthermore one has possibly different actor models $\{AM_{exec}, AM_{assist}, ...\}$, and one has a demonstrator $Demo$ with the best interface (D_i, V^n). Between the assistive and the executive actor model exists a logical dependency as well as between all actor models and the actor story: without the actor story the actor models are underspecified. That means the *whole specified behavior* M_{SR} is only given as the complex structure $\langle AS, AM_{exec}, AM_{assis}, \iota_{as,am-exec}, \iota_{as,am-assis}\rangle$ where the mappings ι connect the actor story with the embedded models.

6.1 Logical Design, Implementation, Validation

To *convert* these results *into a real working system* SYS_{assis} one has to process[9] a *logical design phase* δ which takes into account the whole specified behavior M_{SR} as requirements for the behavior of the intended system. The outcome should be a *blue-print* $M_{SR,design}$ for the implementation of a real system, written as

$$\delta \quad : \quad M_{SR} \longmapsto M_{SR,design} \tag{8}$$

[8] It is actually not clear how 'big' this n should be. Some research is needed.

[9] For all assumed phases in a systems engineering process see formula 1 in section 1 and more elaborated in the paper Erasmus & Doeben-Henisch 2011 [EDH11a]

Based on such a blue-print the *implementation phase* σ translates these ideas in a physical entity $M_{SR,real}$, written as

$$\sigma \quad : \quad M_{SR,design} \longmapsto M_{SR,real} \tag{9}$$

Because the transfer from the AAI-analysis phase into the logical design phase as well the transfer from the logical design phase into the implementation phase can principally not completely be defined one has to run a *validation phase* υ_υ which compares the *behavior requirements* M_{SR} from the AAI-analysis phase with the *behavior* of the *real system* $M_{SR,real}$. The outcome will be some percentage of agreement with the required behavior, written as

$$\upsilon_\upsilon \quad : \quad M_{SR} \times M_{SR,real} \longmapsto [0,1] \tag{10}$$

6.2 Conceptual Gap In Systems Engineering?

The theoretically required validation of the behavior of the real system $SYS_{assis,real}$ with the required behavior specified as whole behavior model M_{SR} can not work out directly, as long as the specified behavior is not available in some *implemented* format.

Diverging from the usual processing of systems engineering it will be assumed in this text that the whole specified behavior M_{SR} will be translated into a blue-print within logical design (cf. Formula 8) and similarly will the blue-print version of the whole behavior $M_{SR,design}$ completely be converted in a real version $M_{SR,real}$ including not only the intended assistive actor but also the complementary executive actor as well as the necessary actor story (cf. Formula 9).

One way to realize this concept is to implement real simulators to mimic the required behavior . Especially it should be possible that real users can take over the role of the simulated executive actors within such simulations or the real world is another actor which takes over the role of the simulated world of the simulated actor story.

7 The AASE-Paradigm

The text so far gives only a very limited account of the whole *Actor-Actor Systems Engineering (AASE)* paradigm. We hope to be able to develop it further with many illustrating applications (case studies).

Everybody is invited to share the discussion of this new paradigm with questions, critical remarks, hints, examples, whatever helps to clarify this paradigm.

There exists a minimal project plan to finalize these ideas in a first booklet (theory and case studies) until April 2018 with a publication in May 2018. Then everything can happen.

References

[DH10] Gerd Doeben-Henisch. *Formal Specification and Verification: Short Introduction*. Gerd Doeben-Henisch, http://www.uffmm.org/science-technology/single/themes/computer-science/personal-sites/doeben-henisch/FSV/THEORY/fsv/fsv.html, 2010.

[DHW07] G. Doeben-Henisch and M. Wagner. Validation within safety critical systems engineering from a computation semiotics point of view. *Proceedings of the IEEE Africon2007 Conference*, pages Pages: 1 – 7, 2007.

[EDH11a] Louwrence Erasmus and Gerd Doeben-Henisch. A theory of the system engineering management processes. In *9th IEEE AFRICON Conference*. IEEE, 2011.

[EDH11b] Louwrence Erasmus and Gerd Doeben-Henisch. A theory of the system engineering process. In *ISEM 2011 International Conference*. IEEE, 2011.