...THE DESIGNER OF A NEW SYSTEM MUST NOT ONLY BE THE IMPLEMENTOR AND THE FIRST LARGE-SCALE USER; THE DESIGNER SHOULD ALSO WRITE THE FIRST USER MANUAL...IF I HAD NOT PARTICIPATED FULLY IN ALL THESE ACTIVITIES, LITERALLY HUNDREDS OF IMPROVEMENTS WOULD NEVER HAVE BEEN MADE, BECAUSE I WOULD NEVER HAVE THOUGHT OF THEM OR PERCEIVED WHY THEY WERE IMPORTANT.

DONALD E. KNUTH
Gerd Doeben-Henisch, Louwrence Erasmus, Zeynep Tuncer

Actor Interaction (AAI) within a Systems Engineering Process (SEP)

An Actor Centered Approach to Problem Solving

Version 22. June 2018

Uffmm.org
Contents

Preparing the Viewpoint 15

Actor-Actor Analysis 19

Actor Story (AS) 21

Actor Model (AM) 31

AS-AM Summary 41

Optimal Design 45

After AAI-Analysis 49

Looking Forward; ToDos 51

Bibliography 53

Index 55
**List of Figures**

1. Engineering process with different kinds of actors 16
2. Basic Mappings between empirical reality with body as a part and internal mental structures 22
3. Change event between two states 25
4. Change event with an embedded actor 29
5. Electronic door example - bare graph, only nodes 37
6. aai-example electronic door: nodes and minimally labeled edges 37
7. aai example electronic door with nodes, edge-labels, and properties 37
8. aai example with a complete graph (only the edge labels are shortened) 37
9. Graph with complete start state followed by difference states based on labelled edges 38
10. Summary of Actor Story and Actor Model with Simulators 41
11. Summary of Actor Story and Actor Models connected with the real world of bodies 42
List of Tables
Dedicated to those who gave us the prior experience and the inspiring ideas to be able to develop the view offered in this book.
Introduction

This book is our first trial to bring together such diverse topics like Human-Machine Interaction, Systems Engineering, Philosophy of Science, and Artificial Intelligence.

The text points back to the paper "AAI - Actor-Actor Interaction. A Philosophy of Science View" from 3.Oct.2017 and version 11 of the paper "AAI - Actor-Actor Interaction. An Example Template" and it 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 (AAI) or even the older Human-Computer Interaction (HCI) is now embedded within the new AASE approach. The same holds for the fuzzy discipline of Artificial Intelligence (AI) or the subset of AI called Machine Learning (ML). Although the AASE-approach is completely in its beginning one can already see how powerful this new conceptual framework is.

Additionally there exists 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 \(^1\) with ongoing discussions since then, a lecture by Doeben-Henisch about formal specification and verification in 2010 \(^2\), two papers by Erasmus and Doeben Henisch in 2011 \(^3\), 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 AAI 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 AAI since 2015.
Preparing the Viewpoint

**History: From HCI to AAI**

To speak of ‘Actor-Actor Interaction (AAI)’ instead of ‘Human-Computer Interaction (HCI)’ is rooted in the course of history. When the World War II ended several advances in technology and software gave raise to great expectations and visions what the future can bring mankind to improve life.¹

Looking to the course of events between 1945 and about 2000 one can observe a steady development of the hardware and the software in many directions. This caused an explosion in many variants of new applications and usages of computer. This continuous challenge of how human persons can interact with this new technology provoked a rapid development what has been called in the beginning ‘Human Computer Interaction (HCI)’. But with the extension of the applications in nearly all areas of daily live from workplace, factory, to education, health, arts and much more the interaction was no longer restricted to the ‘traditional’ computer but interaction happened with all kinds of devices which internally or in the background used computer hardware and software. Thus a ‘normal’ room, a ‘normal’ street, a ‘normal’ building, a toy, some furniture, cars, and much more turned into computerized devices with sensors and actuators. At the same time the collaborators of human persons were not only other human persons or certain animals but more and more ‘intelligent’ machines, robots, smart interfaces. Thus to speak of a ‘human user’ interacting with a ‘technical interface’ was no longer appropriate. A more appropriate language game is the new talk of ‘interacting actors’, which can be sets of different groups of actors interacting in some environment to fulfill a task. Actors are then biological systems (man as well as animals) and non-biological systems.

¹ For some ‘bits of history’ see Doeben-Henisch (2018)
Engineering: Different Views

If one wants to deal with the development of optimal interfaces within certain tasks for executing actors one can distinguish different views onto this problem (see figure 1).

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:

$$SEP(x) \iff x = (P,S, Sep)$$

$$\begin{align*}
Sep & : P \rightarrow S \\
Sep & = \alpha \otimes \delta \otimes \mu \otimes \nu \otimes o \\
\alpha & : \text{Analysis of the problem P} \\
\delta & : \text{Logical design} \\
\sigma & : \text{Implementation of S} \\
\nu & : \text{Validation} \\
o & : \text{Deployment}
\end{align*}$$

Figure 1: Engineering process with different kinds of actors

\textsuperscript{5} Today still mostly human persons.

\textsuperscript{6} For the first paper of Erasmus together with Doeben-Henisch about this subject see

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.

**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. These considerations are done in a separate paper called ‘Philosophy of the AAI-Expert’ (see Doeben-Henisch (2018) 7).

The main topic in the philosophy paper is centered around the findings of modern biology and psychology that the ability of human persons to use a set theoretical language $L_c$ to talk about the experiences with the world is grounded in the cognitive machinery of human persons including complex processes related to perception, memory, spatial and temporal thinking, embedding of languages and others. Because the human brain in the body is not directly interacting with the outside world but mediated by sensors and actuators it is this complex cognitive machinery which constructs an inner model of the outside world. And it are exactly the properties of this ‘inner model’ which provide a ‘point of reference’ for all our thinking and talking.

One conclusion from these considerations is that the reality for a human person is basically perceived as a stream of events, which can be divided in distinguishable situations, called states. A state is understood as a set of properties embedded in a three-dimensional space. If at least one property changes a state changes. Subsets of properties can be understood as objects, which in turn can be subdivided into ‘actors’ and ‘non-actors’. Actors can ‘sense’ their environment and they can ‘respond’. More distinctions are possible as needed.

To understand how an AAI-expert perceives his world, generates internal models, and how he is communicating with others, one has to clarify these philosophical groundings.
Actor-Actor Analysis

After these introductory remarks we start with the Actor-Actor analysis (AAI-analysis) within the systems engineering process. One has to provide a problem to be solved as initial step.

Problem to be Solved

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. In this context it suffices to describe shortly in the introduction of the problem document which persons have been participating in the communication with their communication addresses for further questions.

2. Due to the fuzziness of human communication one has to assume to a certain degree a semantic gap with regard to the participants of the communication which generated the problem document as well as for potential readers of the problem document.

3. Additionally to the general problem a finite set of special constraints (C) can be given, which correspond to the traditional ‘non-functional requirements’. To do this in the right way one has to describe the ‘intended meaning’ of these constraints in a way that it is possible either to decide, whether this intended meaning is fulfilled by the following actor story and actor models or that these constraints pointing to the follow up phases of the systems engineering process.

Check for AAI-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.

---

For an early discussion of one of the authors about the semantic-gap problem see Doeben-Henisch & Wagner (2007).

AAI-Analysis

The goal of the AAI-analysis is to find an optimal assistive actor (AssA)\(^9\) to support the executive Actor (ExecA)\(^10\) 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 simulation. Actually it is not clear whether one should prefer the sequence textual – pictorial – formal – simulation or textual – formal – pictorial – simulation. Below the first sequence is used.,

\(^9\) Traditionally understood as the technical interface.
\(^10\) Traditionally understood as the human user.
**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 3rd-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 3rd-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 3rd-person view one has to construct additional models called Actor Models (AMs). For more details have a look to the section ??.

The relationship between the traditional ‘functional requirements (FR)’ and the ‘actor story’ is such, that all necessary functional requirements have to be part of the actor story. The ‘non-functional requirements (NFR)’ have to be defined in their intended meaning before the actor story and then it must be shown, how the structure of the actor story ‘satisfies’ these criteria,. In this sense are the ‘non-functional requirements’ presented as ‘constraints’ which have the status of ‘meta-predicates’, which have to be designed in an appropriate ‘control logic’ for actor stories.

The topic of ‘Non-Functional Requirements (NFRs)’ as well as ‘Functional Requirements (FRs)’ and their relationship is a hot topic in systems engineering and has not yet a complete solution. The general problem is how to ‘represent’ the NFRs in a way, that these can be handled in the overall system. The big advantage of the AASE paradigm in this context is that the mathematical version of the actor story provides a formal structure which allows to describe all functional requirements (FRs) in a formal way which allows the annotation of non-functional requirements (NFRs) easily.

The Philosophy behind the Actor-Story concept – as pointed out in the figure 2 – is given in a draft paper by Doeben-Henisch (2018) describing the basic relationships between the empirical external world with the body as a part and the internal, mental, structures and processes enabling things like concepts, memories, languages with meaning etc.

From this one can derive that different modes to represent empirical matters with symbolic expressions like a language $L$ have as primary point of reference the ‘mental ontology’ $DAT_{intol}$

---

of the AAI experts. While the mental ontology is assumed to be ‘the same’ for all different modes of symbolic articulation\(^\text{12}\), the different modes of articulation can express different aspects of the same mental ontology more highlighted than in other modes of symbolic articulation.

In the case of expressions of some ‘everyday language’ \(L_0\) like German or English we have only symbols of some alphabet, concatenated to strings of symbols or articulated as a stream of sounds. Thus an understanding of the intended meaning is completely bound to the mental encoding of these expressions, eventually associated with some other clues by body-expressions, mimics, special contexts, and the like.

If we would use a ‘pictorial language’ \(L_{\text{pict}}\) as in a comic strip, we would have again some strings of symbols but mostly we would have sequences of two-dimensional drawings with the symbols embedded. These drawings can be very similar to the perceptual experience of spaces, objects, spatial relations, timely successes, and more properties which somehow ‘directly’ encode real situations. Thus the decoding of the symbol expressions is associated with a strong ‘interpretation’ of the intended situations by ‘world-like pictures’. In this sense one could use such a pictorial language as a ‘second hand ontology’ for the encoding of symbolic expressions into their intended meaning.

But for the intended engineering of the results of an AAI analysis neither the everyday language mode \(L_0\) nor the pictorial language mode \(L_{\text{pict}}\) is sufficient. What is needed is a ‘formal language’

\(^{12}\) Which is a highly idealistic assumption in case of learning systems.
can easily be used for logical proofs, for automated computations, as well as for computer simulations. One good candidate for such a formal language is a language using mathematical graphs which are additional enriched with formal expressions for properties and changes between states. This allows an automatic conversion into automata which can simulate all these processes. Additional one can apply automatic verification for selected properties, e.g. for non-functional requirements!

From this we derive the following main modes of an actor story in this text: (i) Everyday language \( L_0 \) (here English), (ii) Pictorial language \( L_{\text{pict}} \) (in this version of the text not yet defined), (iii) Formal language \( L_{\epsilon} \), (iv) Converted automaton \( a_{L_\epsilon} \) out of the formal language, which can simulate the actor story.

The additional actor models described after the actor story can be seen as special extensions of the actor story and have to be included in the simulation mode. This is straightforward but has also not yet been included in this version of the text.

**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 \( L_{\text{EN}} \). 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 subdivided 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 \( A_{\text{assist}} \) are those who are expected to support the executive actors \( A_{\text{exec}} \) in fulfilling some task \( t \) (with \( t \in T \)).

A task is assumed to be a sequence of states with a start state \( s_{\text{start}} \) and a goal state \( s_{\text{goal}} \), 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 the goal state. The states between the start and the goal state are called intermediate states.

Every finished actor story has at least one path.\(^{13}\)

**Pictorial Actor Story (PAS)**

In case of an textual actor story (TAS) – as before explained – one has a set of expressions of some common language \( L_0 \). These

\(^{13}\) 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.
expressions encode a possible meaning which is rooted in the internal 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 ‘Philosophy of the View-Point’ (and in the figure 2) 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 3rd-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: \( \pi : L_o \leftrightarrow L_{pict} \).

**Mathematical Actor Story (MAS)**

To translate a story with spatial structures and timely changes into a mathematical structure one can use a mathematical graph \( \gamma \) extended with properties \( \Pi \) and changes \( \Xi \) for encoding.

A situation or state \( q \in Q \) given as a spatial structure corresponds in a graph \( \gamma \) to a vertex (also called ‘node’) \( v \), and a change \( \xi \in \Xi \) corresponds to a pair of vertices \( (v, v') \) (also called 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 \( \lambda : V \mapsto 2^{L_{\Pi}} \) and every edge \( e \in E \) into a set of change-expressions \( L_{\xi} \) with \( \epsilon : E \mapsto 2^{L_{\xi}} \) then a vertex in the graph \( \gamma \) 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 \( \gamma \) extended with properties and changes is called an extended graph \( \gamma^+ \).

Thus we have the extended graph \( \gamma^+ \) given as:

\[
\gamma^+(g) \quad \text{iff} \quad g = \langle V, E, L_{\Pi}, L_{\xi}, \lambda, \epsilon \rangle
\]  

(2)
Figure 3: Change event between two states

The occurrence of a change is represented by two vertices \( v, v' \) connected by an edge \( e \) as \( e : \{v\} \mapsto \{v'\} \). 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 \( \epsilon \in L_\lambda \) containing a list of properties to be deleted as \( d : \{p_1, p_3, \ldots\} \) and properties to be newly created as \( c : \{p_2, p_4, \ldots\} \).

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

These conventions define the actor story as formal mathematical graph enhanced by formulas form properties and formal expressions for changes.

Objects and Actors 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 \( A \subseteq OBJ \).

1. Generally it is assumed that there exists some ‘domain of reference’ \( D^R \) which corresponds to a situation/ state of an actor story.

2. For every ‘object’ in \( D^R \) one can introduce a ‘name’ realized as a string of ‘small alphanumeric letters’ beginning with a ‘capital letter’. Names are a subset of terms. Examples: ‘Hobbes’, ‘U2’, ‘Moon’, ...
3. Mappings from distinct objects into other distinct objects which have all to be objects of $D^R$ are called ‘functions’ realized as a string of ‘small alphanumeric letters’ followed by n-many terms enclosed in brackets. Functions are as well a subset of terms. Examples: ‘add(3,4)’, ‘push(Button1)’, ...

4. ‘Properties’ Π are relations between objects in an assumed ‘domain of reference’ $D^R$. The properties are symbolically represented by property expressions $L_\Pi$ which are realized by n-many terms functioning as ‘arguments’ of n-ary ‘predicates’. Thus a property-expression is a sequence of an n-ary ‘predicate’ as a string of ‘big alphanumeric letters’ enriched with the ‘‘-sign followed by n-many terms as arguments enclosed in brackets. Example: ‘USER(U1)’, ‘SCREEN(S)’, BUTTON(B1)’, ‘IS-PART-OF(B1,S)’, ‘ON(push(B1))’, ...

5. As stated above there exists a mapping from states into sets of property expressions written as $\lambda : V \rightarrow 2^{L_\Pi}$

1. A change in the domain $D^R$ happens when at least one property disappears or emerges. To express this symbolically one has to assume (as stated above) that there are two formal states $v, v'$ each with property expressions $L_{\Pi v}, L_{\Pi v'}$ and the property expressions from follow-up state $v'$ are generated by applying a ‘change-action’ realized as a function $a \in ACT$ to the preceding state $v$. The change action has a ‘name’ realized by a string of ‘small alphanumeric values’ followed by a ‘delete function’ named ‘delete’ (or short ‘d’) and then by a ‘creation function’ named ‘create’ (or short ‘c’). Thus the change action $a$ is a concatenated operation $a = d() \otimes c()$. The arguments of the delete- and create-function are property expressions.

2. Example: if there is a set of property expressions $L_{\Pi v} = \{SCREEN(S), BUTTON(B1), NOT-PRESSED(B1)\}$ and a change action $a(L_{\Pi v})$ with the sub-functions $d(NOT-PRESSED(B1))$ and $c(PRESSED(B1))$ then the resulting follow-up property set looks like $L_{\Pi v'} = \{SCREEN(S), BUTTON(B1), PRESSED(B1)\}$

3. The complete change expression will be realized as a ‘list’: $\langle v, v', a, d(p_1, ..., p_n), c(p_1, ..., p_m) \rangle$. This reads: a change action with name $a$ has been applied to state $v$ and generates a new state $v'$ by (i) copying the properties from state $v$ to state $v'$, then (ii) deletes the properties $(p_1, ..., p_n)$ in $v'$, and then (iii) creates the properties $(p_1, ..., p_m)$ in $v'$. The result of applying (i) - (iii) to the old state $v$ generates the new state $v'$.

4. Thus change statements are terms derived as a subset as follows: $\epsilon \subseteq V \times V \times ACT \times \Pi^{Nat} \times \Pi^{Nat}$ (with Nat as the natural numbers including 0).

5. If there is in one state $v$ more than one action possible than more than one change statement is possible. This results in more than one edge leading from state $v$ to n-many follow-up states $v'_1, ..., v'_n$.

4 The default assumption is that either the delete or the create function has to have at least one property argument.
6. Additional to the names of possible objects we assume a special operator ‘not($n$)’ applied to a name ‘$n$’. The meaning of the operator is, that in this case not the name ‘$n$’ is valid, but the ‘absence’ of the object signified by the name $n'$. This is important because otherwise in case of many alternative options one has to enumerate all alternatives to an object named ‘$n$’.

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

Simulated Actor Story (SAS)

A simulated actor story (SAS) corresponds to a given extended graph $\gamma^+$ by mapping the extended graph into an extended automaton $\alpha^+$.

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. $\Sigma$ as a finite input alphabet
5. $\Delta \subseteq Q \times \Sigma^* \times Q$ 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. $L_{\chi}$ 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 $L_{\Pi}$ as follows: $\langle V, I, F, L_{\chi}, L_{\Pi}, E, \lambda \rangle$ with $\lambda : V \rightarrow 2^{L_{\Pi}}$.

With this definition one has an extended automaton $\alpha^+$ 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 $L_{\Pi}$. All state-transitions of the automaton $\alpha^+$ from a start-state to a goal-state are called a run $\rho$ of the automaton. The set of all possible runs of the automaton is called the execution graph $\gamma_{exec}$ of the automaton $\alpha^+$ or $\gamma_{exec}(\alpha^+)$. 
Thus the simulation of an actor story corresponds to a certain run $\rho$ of that automaton $\alpha^+$ which can be generated out of a mathematical actor story by simple replacement of the variables in the graph $\gamma^+$.

**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 $A^+$ \( R\text{Prof}_A \). 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.

**Actor Induced Actor Requirements (AAR)**

Because the required profile $R\text{Prof}_{requ}$ 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 $R\text{Prof}_{avail}$ is either in a sufficient agreement with the required profile or not, $\sigma : R\text{Prof}_{requ} \times R\text{Prof}_{avail} \mapsto \{0, 1\}$.

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 $\chi : A_{avail,exec} \times R\text{Prof}_{requ} \mapsto A_{requ,exec}$. If such an improvement is not possible then the planned task cannot be realized with the available executing actors.

**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 \( \text{Int}_{F_{\text{assis}}} \), 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 \( \text{Int}_{F_{\text{assis}}} \) 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.

**Actor Model and Actor Story**

While one can describe in an actor story (AS) possible changes seen from a 3rd-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.

The general idea of this interaction between actor story and actor model can be seen in figure 4.

1. In a simple actor story with only two states \( v, v' \) we have an actor called ‘USER(U1)’ which has ‘visual perception’ and which can act with ‘motor activities’.

2. Therefore the actor can ‘see’ properties like ‘SCREEN’, ‘BUTTON’, and ‘NOT-PRESSED’. Based on its ‘behavior function’ \( \Phi \) the actor can compute a possible output as a motor-action, described as an event expression \( \langle v, v', \text{press}(\text{BUTTON}(B1)), \text{d(not-pressed}(B1)), c : (\text{pressed}(B1)) \rangle \).

3. This results in a change leading to \( v' \). The actor \( U1 \) is left out in \( v' \), also it is still part of \( v' \).
**Actor Model (AM)**

Seen from the actor story the processing of the task requires that an actor can *sense* all necessary aspects of the task processing as well as he can *respond* as needed. Besides this one expects that the actor is able to *process* the *input information* \((I)\) in a way that the actor is able to *generate* the right *Output* \((O)\). One can break down the *required behavior* to a series of necessary inputs \(I\) for the actor followed by necessary responses \(O\) of the actor. This results in a series of input-output pairs \(\{(i,o),\cdots,(i,o)\}\) defining implicitly a required *empirical behavior function*:

\[
\phi_e = \{(i,o),\cdots,(i,o)\} \quad (6)
\]

Because any such empirical behavior function is finite and based on single, individual events, it is difficult to use this empirical finite function as the function of an explicit model. What one needs is an explicit *general theoretical behavior function* like:

\[
\phi: I \rightarrow O \quad (7)
\]

Although an empirical behavior function \(\phi_e\) is not a full behavior function, one can use such an empirical function as a *heuristic guide* to construct a more general theoretical function as part of a complete hypothetical model of the actor.

It is an interesting task, to *elaborate a hypothetical model* of the internal processes of an actor which defines the *theoretical behavior function* \(\phi\). To do this broadly with all details is beyond the scope of this text. Instead we will work out a first basic model which can be understood as a kind of a *template* for theoretical behavior functions, which can be extended further in the future.

The task of modeling a possible actor is twofold: first (i) one has to define a complete *formal model* of a possible structure and it's dynamic, second (ii) it must be possible to *predict the behavior* of the model in a way that it is possible to *observe* and *measure* this behavior. If the observable behavior of the model is *including* the empirical behavior function \(\phi_e\), then the hypothetical model is *empirical sound in a weak sense*.

\[
\phi_e \subseteq \phi \quad (8)
\]
We understand here a model as a mere collection of rules, while an algebraic structure is an extension of a model by including additional sets as well as axioms. But we use the term ‘model’ here equivalently to the term ‘algebraic structure’.

**Actor as Input-Output System**

To enable 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 (i) inputs (I) from the environment (here the actor story), which are translated by some kind of a ‘sensoric system’ generating inputs (I) of this environment as well as (ii) outputs (O) which can cause changes in the environment. The sum of all inputs I and outputs O defines the basic interface (BIntf) of an input-output system S in an environment E, written \( BIntf_{S,E} = \{x|x \in I \times O\} \). We can write this as follows:

**Def: Input-Output System (IOSYS)**

\[
IOSYS(S) \iff S = \langle I, O \rangle \\
I := \text{Input} \\
O := \text{Output}
\]

(9)

and:

**Def: Basic Interface (BIntf)**

\[
E := \text{Environment of system } S \\
IOSYS(S) \iff S = \langle I, O \rangle \\
I := \text{Input} \subseteq E \\
O := \text{Output} \subseteq E \\
BIntf_{S,E} = \{x|x \in I \times O\}
\]

(11)

(12)

**Def: Real Interface (RIntf)**

The basic interface (BIntf) has to be distinguished from that interface which represents a ‘real’ device interacting with an executive actor. The real interface (RIntf) of an assistive actor ‘realizes’ the basic interface by providing some sensoric appearance of an assistive actor. Thus if the executive actor needs an input from the interface there can be visual or acoustic or haptic or other sensoric properties which are used to convey the input to the executive actor. As well, if the executive actor wants to produce an output to change some properties in the assistive actor there must be some sensor at the side of the assistive actor which can receive some ‘action’ from the executive actor. The concrete outlook of such a real interface is the task of the ‘interface design’ given a ‘basic interface’.
Learning Input-Output Systems

A ‘learning input-output system (LIOSYS)’ is a special case of an input-output system because its ‘basic interface can change’! This dynamic behavior is described by a ‘learning behavior function’ \( \phi \). The learning behavior function assumes additionally to the inputs (I) and outputs (O) a non-empty set of ‘internal states (IS)’ which can change their properties. These dynamic internal states can ‘represent’ some properties of the inputs in a way, that the system can take these properties into account for the ‘computation of the next output’. Depending on the ‘quality’ of these internal representations the system can pre-view ‘possible states’ of the environment and its own situation in it. This structure of internal states as part of a behavior function is usually called ‘memory’ \( \mu \).

A most general definition of learning input-output systems can be given as follows:

\[
\text{Def: Learning Input-Output System (LIOSYS)}
\]

\[
\text{LIOSYS}(S) \iff S = \langle I, O, IS, \phi \rangle
\]

\[
\begin{align*}
I & : \text{Input} \\
O & : \text{Output} \\
IS & : \text{Internal states} \\
\phi & : I \times 2^{IS} \rightarrow 2^{IS} \times 2^O
\end{align*}
\]

From this it follows that the ‘basic interface’ is only a subset of the behavior function of a learning system. This means to ‘understand a learning input-output system’ it is not sufficient to describe the behavior of a system only once; instead one has to describe the behavior in different phases to detect ‘possible changes’ compared to the ‘past’. This corresponds to the fact, that a learning system ‘learns always’. Thus to ‘predict’ the behavior of learning systems in an environment is in no case trivial.

The change of behavior which is a property of learning systems is usually driven by ‘random’ as well as ‘non-random’ properties. Thus to describe the ‘dynamics’ resulting from a learning behavior one needs a sufficient knowledge about these ‘change-inducing properties’. In real biological systems these change-inducing properties seem to be composed of bundles of different functions interacting in a non-trivial way with each other. Until today a complete and transparent description of these change-inducing properties seems by far to be a too complex task to do.

**Empirically and Non-Empirically Motivated**

The general definition of a learning input-output (see definition: 13) offers space for nearly infinite many concrete instances. One possible classification scheme could be that of empirically motivated or non-empirically motivated models.
Empirically Motivated Examples of empirically motivated models are some of the models which experimental psychologists have tried to develop. One famous team of psychological motivated researchers was the team Card, Moran and Newell working at the Paolo Alto Research Center (PARC) starting in 1974. They published a book ‘The Psychology of Human-Computer Interaction’ where they showed how one can develop empirical models of human actors. According to Card et al.(1983) one can assume at least three sub-functions within the general behavior function:

\[
\phi = \phi_{perc} \otimes \phi_{cogn} \otimes \phi_{mot}
\]  

\[\phi_{perc} : I \rightarrow (VB \cup AB) \]  

\[VB : = \text{Visual buffer} \]  

\[AB : = \text{Auditory buffer} \]  

\[\phi_{cogn1} : (VB \cup AB) \times M_{STM} \rightarrow M_{STM} \]  

\[\phi_{cogn2} : M_{STM} \times M_{LTM} \rightarrow M_{STM} \times M_{LTM} \]  

\[\phi_{mot} : M_{LTM} \rightarrow O \]  

Thus an input – visual or auditory – will be processed by the perception function \(\phi_{perc}\) into an appropriate sensory buffer \(VB\) oder \(AB\). The contents of the sensory buffers will then be processed by the partial cognitive function \(\phi_{cogn1}\) into the short term memory (STM), which at the same time can give some input for this processing. Another cognitive function \(\phi_{cogn2}\) can map the contents of the short term memory into the long term memory (LTM) thereby using information of the long term memory as input too. From the long term memory the motor function can receive information to process some output \(O\).

According to these assumptions we have to assume the following partitions of the internal states:

\[VB \cup AB \cup M_{STM} \cup M_{LTM} \subseteq IS\]

The complete model can be found in the cited book.

Non-Empirically Motivated In many cases non-empirically motivated models are sufficient. This amounts to the task to ‘invent’ a function \(\phi\) which maps the inputs from the known actor story into the outputs of the known actor story. This can be done deterministically or non-deterministically, i.e. in a learning fashion.

In the deterministic case one can take the empirical behavior function (see definition 6) derived from the actor story ‘as it is’.

In the non-deterministic case it is not enough to ‘re-write’ the empirical behavior function as the theoretical behavior function.
of the actor model. To adapt to the documented changes in the behavior of the actor one has to assume ‘appropriate’ internal states whose internal changes correspond to the observable changes in the actor story.

**GOMS Model**

One old and popular strategy for non-empirically motivated models is labeled GOMS for Goals, Methods, Operators and Selection rules.\(^\text{16}\)

- **GOAL**: A *goal* is something to be achieved and will be represented by some *language expression*.
- **OPERATOR**: An *operator* is some *concrete action* which can be done.
- **METHOD**: A *method* is a composition of a goal and some operators following the goal to realize it.
- **SELECTION RULE**: A selection rule has an IF-THEN-ELSE structure: *IF* a certain condition is fulfilled, *THEN* some method will be selected, otherwise the method following the *ELSE* marker will be selected.

According to the general learning function \(^\text{13}\) a rule of a GOMS model has the logical format:

\[
IF I = X \land IS = Y \quad THEN \quad IS = Y' \land O = Z
\quad (25)
\]

**Example: An Electronically Locked Door** For the following demonstration we use the simple example of an electronically locked door.\(^\text{17}\)

For this actor model in the GOMS format we assume the following formal actor story:

**AS for Electronic Door Example** If we start with state Q₁, then it will be followed by state Q₂ if the output of the executive actor is pushing the key with symbol A; otherwise, if the output is different, then we will will keep state Q₁. Similar in the following states: If we are in state Q₂ and the output of the user is pushing the key with symbol B, then the user story switches to state Q₃; otherwise we are back in state Q₁. Finally, if we are in state Q₂ and the user pushes the key with symbol A, then we will reach the final state Q₄, otherwise back again to state Q₁.

The details of the different states are given here.

1. Start = U₁ ∪ S₁ ∪ Env₁
   
   U₁ = \{USER(U₁)\}
   
   S₁ = \{SYSTEMINTF(S₁),KEYPAD(K₁), PART-OF(K₁, S₁), KEY(K₁),
   KEY(K₂), KEY(K₃), PART-OF(Ka, Kb, Kc), K₁\}

\(^\text{16}\) A first extensive usage of a GOMS model can be found in Card et al. (1983) :139ff


\(^\text{17}\) For a description of the example see: http://www.doeben-henisch.de/fh/fsv/node13.html in Doeben-Henisch (2010).

\[ \text{Env}_1 = \{\text{DOOR(D}_1), \text{CLOSED(D}_1)\} \]

Meaning: ‘U\(_1\)’ is the name of a user, ‘S\(_1\)’ the name of a system-interface, and ‘Env\(_1\)’ is the name of an environment. All three ‘U\(_1\), S\(_1\), C\(_1\)’ are names for subsets of properties of state Start.

2. CHANGE-AS: \(\langle \text{Start,Start,} \text{push(not(Ka),K}_1\rangle, d()\rangle, \langle \text{Start,Q}_2, \text{push(Ka,K}_1\rangle, d(), c(\text{PRESSED(Ka)})\rangle)\),

3. \(\text{Q}_2 = \text{U}_1 \cup \text{S}_1 \cup \text{Env}_1\)
   \(\text{U}_1 = \{\text{USER(U}_1\}\}
   \text{S}_1 = \{\text{SYSTEMINTF(S}_1), \text{KEYPAD(K}_1), \text{PART-OF(K}_1, \text{S}_1\}, \text{KEY(Ka), KEY(Kb), KEY(Kc), PART-OF(\langle Ka, Kb, Kc \rangle, K}_1), \text{PRESSED(Ka)}\}\)
   \(\text{Env}_1 = \{\text{DOOR(D}_1), \text{CLOSED(D}_1)\} \)

4. CHANGE-AS: \(\langle \text{Q}_2, \text{Start,} \text{push(not(Kb),K}_1\rangle, d()\rangle, \langle \text{Q}_2, \text{Q}_3, \text{push(Kb,K}_1\rangle, d(), c(\text{PRESSED(Kb)})\rangle)\),

5. \(\text{Q}_3 = \text{U}_1 \cup \text{S}_1 \cup \text{Env}_1\)
   \(\text{U}_1 = \{\text{USER(U}_1\}\}
   \text{S}_1 = \{\text{SYSTEMINTF(S}_1), \text{KEYPAD(K}_1), \text{PART-OF(K}_1, \text{S}_1\}, \text{KEY(Ka), KEY(Kb), KEY(Kc), PART-OF(\langle Ka, Kb, Kc \rangle, K}_1), \text{PRESSED(Kb)}\}\)
   \(\text{Env}_1 = \{\text{DOOR(D}_1), \text{CLOSED(D}_1)\} \)

6. CHANGE-AS: \(\langle \text{Q}_3, \text{Start,} \text{push(not(Ka),K}_1\rangle, d()\rangle, \langle \text{Q}_3, \text{Goal,} \text{push(Ka,K}_1\rangle, d(\text{CLOSED(D}_1), c(\text{PRESSED(Ka), OPEN(D}_1)\rangle)\)

7. Goal = \(\text{U}_1 \cup \text{S}_1 \cup \text{Env}_1\)
   \(\text{U}_1 = \{\text{USER(U}_1\}\}
   \text{S}_1 = \{\text{SYSTEMINTF(S}_1), \text{KEYPAD(K}_1), \text{PART-OF(K}_1, \text{S}_1\}, \text{KEY(Ka), KEY(Kb), KEY(Kc), PART-OF(\langle Ka, Kb, Kc \rangle, K}_1), \text{PRESSED(Ka)}\}\)
   \(\text{Env}_1 = \{\text{DOOR(D}_1), \text{OPEN(D}_1)\} \)

For a complete representation as a graph different variants have been realized to enable a better judgment about the Pros and Cons of the different versions.

The graphs are constructed with the DOT-Language using a normal editor under Linux and the KGraphViewer program based on the graphviz package of software tools developed since 1991 by a team at the ATT&Laboratories. For the theory see e.g. Gansner et.al (1993) \(^8\), and Gansner et.al. (2004) \(^9\). For a tutorial see Gansner et.al (2015) \(^{20}\).

For practical reasons it seems that the last version, figure 9, should be preferred: it gives implicitly all necessary informations and keeps the amount of written information low.

**GOMS Actor Model**

As one can see the formal description of the actor story offers no information about the internal structures which determine the behavior of the different users, the executive actor as well as the assistive actor. To enhance this one has to define additional actor models.


Figure 5: Electronic door example - bare graph, only nodes

Figure 6: aai-example electronic door: nodes and minimally labeled edges

Figure 7: aai example electronic door with nodes, shortened edge-labels, and subsets of properties

Figure 8: aai example with a complete graph (only the edge labels are shortened)
We will start the construction of a GOMS model for the executive actor using the electronically locked door. For this we simplify the GOMS-Model format as follows: 

**IF** Input ... **Internal** ... **THEN** ... **Internal** ... Out ... **ELSE** ... **Internal** ... Out ... The Input can either be some value from the set $I$ of possible inputs or from the set $IS$ of the internal states of the system. In the used example are all properties of the states a possible input or the properties of the internal states. All these IF-THEN rules are subsumed under the goal to enter the open door.

1. **GOMS MODEL FOR USER $U_1$**

2. **INPUT** $U_1 = VB$; **OUTPUT** $U_1 = MOT$

3. **GOAL**: OPEN($D_1$) & DOOR($D_1$)

   (a) **IF** $VB='CLOSED(D_1)$' & $IS='MEM(U_1,(Ka,0,Kb,0,Ka,0))'$ **THEN** $IS='MEM(U_1,(Ka,1,Kb,0,Ka,0))'$ & $MOT='push(Ka,K_1)'$

   (b) **IF** $VB='CLOSED(D_1)$' & $IS='MEM(U_1,(Ka,1,Kb,0,Ka,0))'$ **THEN** $IS='MEM(U_1,(Ka,1,Kb,1,Ka,0))'$ & $MOT='push(Kb,K_1)'$

   (c) **IF** $VB='CLOSED(D_1)$' & $IS='MEM(U_1,(Ka,1,Kb,1,Ka,0))'$ **THEN** $IS='MEM(U_1,(Ka,1,Kb,1,Ka,1))'$ & $MOT='push(Ka,K_1)'$

   (d) **IF** $VB='OPEN(D_1)$' & $IS='MEM(U_1,(Ka,1,Kb,1,Ka,1))'$ **THEN** $IS='MEM(U_1,(Ka,0,Kb,0,Ka,0))'$ & $MOT='movesthrough(U_1,D_1)'$

These IF-THEN-Rules follow the general behavior function $\phi : I \times 2^{IS} \rightarrow 2^{IS} \times O$

The system interface $S_1$ has its own GOMS-Model.

1. **GOMS MODEL FOR SYSTEM-INTERFACE $S_1$**

2. **INPUT** $S_1 = K_1$; **OUTPUT** $S_1$ = states of the door (CLOSED, OPEN)

3. **GOAL**: OPEN($D_1$) & DOOR($D_1$)

   (a) **IF** $K_1 = 'KEY-PRESSED(K_1,Ka)'$ & $IS='CODE(C_2,(Ka,0,Kb,0,Ka,0))'$ **THEN** $IS='CODE(C_2,(Ka,1,Kb,0,Ka,0))'$

   (b) **IF** $K_1 = 'KEY-PRESSED(K_1,not(Ka))'$ & $IS='CODE(C_2,(Ka,0,Kb,0,Ka,0))'$ **THEN** $IS='CODE(C_2,(Ka,0,Kb,0,Ka,0))'$

   (c) **IF** $K_1 = 'KEY-PRESSED(K_1,Kb)'$ & $IS='CODE(C_2,(Ka,1,Kb,0,Ka,0))'$ **THEN** $IS='CODE(C_2,(Ka,1,Kb,1,Ka,0))'$
Measuring Dynamic Behavior

If one assumes a ‘learning actor’ then the actor story describes the ‘expected behavior’ as a set of ‘input-output pairs’ for every state and an actor has to be ‘trained to learn’ the ‘expected sets of input-output pairs’.

One can use therefor the defining sequence of input-output tasks to ‘measure’ the ‘intelligence’ of an actor. Running a task the first time one can use the percentage of correctly solved sub-tasks within a certain amount of time as a ‘benchmark’ indicating some measure of ‘intelligence’. (For an introduction into the topic of psychological intelligence measures see e.g. Eysenck (2004) 21, Rost (2009) 22, Rost (2013) 23)

To measure the ‘learning capacity’ of an actor one can use a task to explore (i) how much time an actor needs to find a goal state and (ii) how many repetitions the actor will need until the error rate has reached some defined minimum. (The history of behavioral Psychology provides many examples for such experiments, see e.g. Hilgard et.al. (1979) 24 and a famous experiment with Tolman (1948) 25 using learning curves and error rates). Another measure could be the quality of the storage capacity (memory) by first identifying a maximum of correctness and then (iii) one measures the duration until which the maximum correctness of the memory has again weakened below a certain threshold of accuracy. (The first scientist who did this in a pioneering work was the German Psychologist Ebbinghaus (1848) 26, English translation 27) While some minimal amount of ‘learning time’ is needed by all kinds of systems – biological as well as non-biological ones – only the non-biological systems can increase the time span for ‘not-forgetting’ much, much wider than biological systems are able to do.

Today the biggest amount of executing actors are still biological systems represented by human persons (classified as ‘homo sapiens’), therefore parameters as ‘learning time’, ‘memory correctness’, or ‘memory forgetting time’ are important to characterize the ‘difficulty’ of a task and ways to explore possible settings which make the task difficult. From such a ‘learning analysis’ one can eventually derive some ideas for possible ‘improvements’. From this follows

\[ \text{if } K_1 = \text{KEY-PRESSED}(K_1, \text{not}(K_2)) \text{ and IS=CODE}(C_2,(K_a,1,K_b,0,K_a,0)) \]
\[ \text{then IS=CODE}(C_2,(K_a,0,K_b,0,K_a,0)) \]
\[ \text{if } K_1 = \text{KEY-PRESSED}(K_1,K_a) \text{ and IS=CODE}(C_2,(K_a,1,K_b,1,K_a,0)) \]
\[ \text{then IS=CODE}(C_2,(K_a,1,K_b,1,K_a,1)) \text{ and OUT=OPEN}(D_1) \]
\[ \text{if } K_1 = \text{KEY-PRESSED}(K_1,\text{not}(K_a)) \text{ and IS=CODE}(C_2,(K_a,1,K_b,1,K_a,0)) \]
\[ \text{then IS=CODE}(C_2,(K_a,0,K_b,0,K_a,0)) \]

Thus a complete Process is an interaction between the actor story (AS) and the actor models written as GOMS-Models.

25 Edward C. Tolman. Cognitive maps in rats and men. The Psychological Review. 55(4):189–208, 1948. 34th Annual Faculty Research Lecture, delivered at the University of California, Berkeley, March 17, 1947. Presented also on March 26, 1947 as one in a series of lectures in Dynamic Psychology sponsored by the division of psychology of Western Reserve University, Cleveland, Ohio
that the format of usability tests should be adapted to these newly identified behavior based properties.

On account of the unobservability of the inner states (IS) of every real system it follows that all assumptions about possible inner states as well as about the details of the behavior function $\phi$ 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.
AS-AM Summary

The Logical Space

Looking back to all these new concepts and complex relationships the figure 10 may be of some help to get the whole picture at once.

1. The AAI experts begin their work with a problem document delivered by some stakeholder.

2. The stakeholder usually is rooted in some part of the real world from where he receives his inspiration for the problem and it is his way of understanding the world and his language which encodes the problem into a problem document $D_p$.

3. The AAI experts analyze the problem by developing in a first phase an actor story (AS) which includes all the circumstances and all the properties which are intended by the stakeholder. The actor story will be realized at least in a textual, in a mathematical, and in a pictorial mode.

4. In a second phase they take the identified actors and develop actor models (AM) to ‘rationalize’ the behavior required by the actor story. This will be done in a formal way.

5. Having a formal description of the actor story as well formal descriptions of the actor models one can directly ‘program’ a
real computer with these specifications. In that case the real
computers are functioning as simulators: the one simulator $\Sigma$
simulating the actor story is representing the actor story is the
world of the simulation, and the different simulators $\sigma_1, \ldots, \sigma_n$
simulating the different actors are actors in this world. The interaction
between the different simulators is realized by message passing.

In this picture of the AAI analysis something important is missing:
all these formalizations and simulations of the actor story and
the different actor models have no defined physical appearance! All
these formalizations are represented as strings of symbols, formal
expressions, even the pictorial language in its strict form. Thus a
grounding in the real world is still missing.

The Physical Space

To give the logical structures of the symbolic space a physical
appearance one has to translate those parts of the logical space into
real things which are necessary for the concrete work.

As you can see in figure 11 there are two kinds of actors which
have to be grounded in the real world of bodies: the assistive actor $A_{ass}$
and the executing actor $A_{exec}$.

While the executing actor in case of human actors has not to be
built but to be recruited from possible candidates, the assistive actor
has to be built as a physical device.

Based on the actor story one can deduce general requirements
for the intended executive actor as 'task induced actor requirements
(TAR)' which state what kinds of inputs the executive actor must
be able to process and what kinds of motor responses. From these
required inputs and outputs one can deduce a basic outline for required cognitive and emotional capabilities. With regard to available candidates one can analyze the capabilities of a real person as actor induced actor requirements (AAR). If the TAR are not in agreement with the AAR then either the candidate is not capable to do the job or he has to be trained to gain the necessary capabilities.

In case of the intended assistive actor there are also logical requirements which can be deduced from the actor story, which describe how the assistive actor should behave. But in this case there exist also additional human-actor based psychological requirements which take into account what a human-actor can perceive and how a human-actor can process perceived information to be able to respond.

It is a special job to create a physical device by obeying these logical and psychological requirements. Until today there is no automatic procedure known to support this.

Because there is no 1-to-1 mapping from the requirements to the physical realization of the assistive actor and no 1-1 mapping from the logical requirements to a real human executive actor it is necessary to organize a series of tests with real human persons using the real assistive actor. Only these tests of usability can reveal, how good the intended interaction of the actors in the intended task works.
Optimal Design

As the preceding section shows, the translation of parts of the logical (symbolical) space into the physical space induces a real amount of fuzziness on both sides, the assistive as well as the executive actor. Therefore one has to realize a series of tests to check the quality of the observable real processes compared to the logical requirement of the actor story.

Usability Measurement Procedure  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. In a usability test UT so far one realizes a mapping of given demonstrators D into a set of usability values V as follows $v_{UT} : D \rightarrow D \times V$. A usability test includes a finite set of objective as well as subjective sub-tests. The values V of one usability test are then given as a finite set of points in an n-dimensional space $V^n$. Thus after a usability test $v_{UT}$ 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 $\mu : 2^{V^n} \rightarrow V^n$ for the assumed finite many n-dimensional values $\{V^n_1, V^n_2, ..., V^n_m\}$ to compare these values and identify for this set an optimal value. Thus $\mu(V^n_1, V^n_2, ..., V^n_m)$ computes a certain $V^n_i \in \{V^n_1, V^n_2, ..., V^n_m\}$.

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.

Not yet Ideally  This is the procedure which is described in most textbooks, but this procedure has a weak point: in these tests one characterizes the test persons as the intended executive actors only roughly, e.g. ‘experienced user’ or ‘normal user’ or ‘beginner’, perhaps additionally one takes into account the ‘age’ and ‘gender’. But as one can infer from the preceding chapters every task has its very specific profile of requirements condensed in the TAR document and what is needed on the executive side is an explicit ‘user profile’ as required with the AAR document. As everybody can easily check a usability test will differ a lot if there are test persons with greatly varying AAR profiles which have different ‘distances’ to the TAR profile. In the extreme case there is a physical assistive device which works fine for test persons with an AAR profile ‘close
to the TAR profile’, but because there haven been test persons with an AAR profile which was ‘not close to a TAR profile’ the results are very bad.

Proposal of an Ideal Procedure  Following the preceding chapters one can infer the following proposal for an ideal test procedure to measure the usability of a physical assistive actor device used by real human persons mimicking the ideal executive actor.

1. Receiving an actor story AS and the TAR document from that story.

2. Selecting a group of test candidates \( \{T_1, ..., T_n\} \) planned to mimicking the intended executive actor.

3. Work out an AAR document for each of the test candidates yielding a set of pairs \( \{(T_1, AAR_1), ..., (T_n, AAR_n)\} \).

4. Compute the distance of each AAR, compared to the TAR and group the test candidates according to their classified actor induced actor requirements AAR into distinct AAR-classes.

5. Run a series of tests and and observe and compute the following for each test:

   (a) Taking notes of the objective behavior data.

   (b) Compare the observed behavior with the expected behavior based on the AS.

   (c) Compute the error rate for each test candidate in each test.

   (d) After one test give the test candidate a questionnaire asking for the general feeling doing the test (-n - o - +n) and asking for objective circumstances connected to this feeling.

6. After the completion of the defined series of tests one has to compute the learning curve for each test person and the curve of satisfaction based on the questionnaires.

7. One continues with another series of tests distributed in time to compute the forgetting curve for each test person not by doing the test but by asking to remember the different action sequences and the test persons are writing down there memories.

With this procedure one can differentiate the different types of test persons more precisely, one will get objective behavior data as well as subjective judgments related to objective properties, and one will get a picture of the dynamic learning behavior of each test person. With these data one can dig ‘deeper’ into the psycho-dynamic of the interaction between human executive actors and physical assistive actors.

If these tests show clear weaknesses within the process of interaction one can try to identify the ‘causes’ for this weaknesses: either (i) physical properties of the assistive actor or (ii) deficiencies on the
side of the executive actors (objectified by the AAR document) or (iii) a bad logic in the actor story.

If the causes seem ‘reasonable’ and their change could improve the overall error rates and the satisfactions in a way which supports the main goal (e.g. earn money with the device, (ii) improve the quality of a service, (iii) improve some theory, ...), then one can decide to improve the actor story or the actor models or the physical device or do a better training for the executing actors.
After AAI-Analysis

Having completed the AAI analysis as pointed out in figure 11 one has to continue in the systems engineering process according to the minimal standard systems engineering process (cf. formula 1) with the logical design phase $\delta$ which takes into account the whole specified behavior with the AS, the AMs, and the TAR, here called $M_{SR} = \langle AS, AM, TAR \rangle$.

Different to the usual analysis phase we have at the end of the AAI analysis phase defined actor models with a mathematical structure which can directly be integrated in the logical design as $M_{SR,design}$ as well we have complete functions which can directly be implemented. Because these functions can be realized with every known programming language this kind of specification can be pre-formatted within the AAI analysis to fit the requirements of the later implementation. Based on such a specified function the implementation phase $\sigma$ translates these ideas in a physical entity $M_{SR,real}$, written as $\sigma : M_{SR,design} \mapsto M_{SR,real}$.

It should be kept in mind that the implementation part of the actor models AM is not primarily intended to be the final solution for implementation but to enable a distributed simulation environment including all actor models $\{\sigma_1, ..., \sigma_n\}$ and the simulator for the whole actor story $\Sigma$ serving as the environment functioning like a ‘world’ in a computer game (cf. figure 11)

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 $\nu_v$ 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

$$\nu_v : M_{SR} \times M_{SR,real} \mapsto [0,1]$$  \hspace{1cm} (26)

Based on the simulated version of the actor story it is possible to realize a validation of the implemented system by coupling the implemented physical system with the simulated actor story as a whole.
Looking Forward; ToDos

The text so far gives only a very limited account of the whole Actor-Actor Interaction (AAI) paradigm within the systems engineering process. We hope to be able to develop it further with many illustrating applications (case studies).

Actual known ToDos  Here is a list of actual ToDos which have to be realized to improve the test:

1. The history part should be refined to make clear to which extend the AAI paradigm has been prepared by different publications, and where have been the differences so far.

2. The different views part should be enriched with more direct comparisons with the main systems engineering positions (especially with INCOSE).

3. The Philosophy of the AAI-Actor part should be extended to give all necessary ideas directly.

4. In the introducing part of the chapter Actor Story one has to explain what non-functional requirements are and it has to be created a new section/ chapter to explain how one can in the AAI-environment give the definition and the check for non-functional requirements in a complete new way.

5. In the subsection of the mathematical AS one has to rewrite all the used formal languages distinguishing between the mathematical graph as such $L_e$, the static-property language $L_\Pi$ attached to the nodes, the dynamic-change language $L_\Xi$ attached to the edges and the specification language of the actor models $L_\alpha$. The domain of reference $D^R$ has to be clarified for all these different languages.

6. It would be great finally to define also the pictorial language $L_{pict}$.

7. Another pending point is the automatic conversion of an AS into a symbolic simulator which then could be implemented on a real machine.

8. It would be great if we could set up a software framework where we can run the AS-simulator $\Sigma$ together with all the actor models $\{\sigma_1, \ldots, \sigma_n\}$; this enhanced with the pictorial language ...
9. It could help to automatize the generation of the TAR document based on the AS.

10. There should be some more theory and a manual based on Psychology which describes how to construct an AAR-document.

11. It has to be defined more clearly how one can compare AAR documents by distances with the TAR document and how to establish a classification based on these comparisons.

12. After rewriting the formal languages for AS and AM the interaction between an AS and the different AMs should be described more precisely.

13. In the chapter about AMs there are many points which can be improved:
   (a) More explanations to the concepts ‘model’ and ‘algebraic structure’ (better perhaps: ‘theory’).
   (b) Generally more examples of actor models with different kinds of formalizations, at least one with a common programming language.
   (c) Providing a small algorithm to support the automatic generation of dot-based-graphs out of the formal description of an AS.

14. Give examples of real usability tests done according to the new protocol.

15. Check the proposed interface between AAI-analysis, logical design, implementation and validation of a systems engineering process with the main mechanisms used today in systems engineering.

   Everybody is invited to share the discussion of this new paradigm with questions, critical remarks, hints, examples, whatever helps to clarify this paradigm. The first address to contact the project is the eJournal: uffmm.org, ISSN 2567-6458, Email: info@uffmm.org. We recommend as start page: https://www.uffmm.org/2017/07/27/uffmm-restart-as-scientific-workplace/
Bibliography


Index

AAI, 15, 16
AAI analysis, 19, 20
AAI-expert philosophy, 17
AAR, 28, 43
actor induced actor requirements, 28
actor model, 31
Actor Story, 21
algebraic structure, 32
AM, 21, 31
AS and AM, 29
AS as graph, 36
AS-AM summary, 41
assistive actor, 20
BIntf, 32
building assistive actors, 42
Card, 34
cardinality, 26
classify AAR, 46
classify AAR, 46
continuous learning, 33
curve of satisfaction, 46
deterministic behavior, 34
domain of reference, 25
dynamics of learning behavior, 46
empirical behavior function, 31
empirically motivated, 33
enhanced graph, 25
environment, 19
everyday language, 22
example electronic door, 35
executive actor, 19
expected behavior, 39
forgetting curve, 46
forgetting time, 39
formal language, 23
functional requirements, 21
functions, 26
fuzzy predictions, 33
generate AAR, 46
generate output, 31
GOAL, 35
GOMS model, 35
GOMS rule format, 35
graph, 24
heuristic guide, 31
HMI-History, 15
human actor, 43
implementation phase, 49
inner states, 40
input-output system, 32
intelligence, 39
interface design, 28, 32
interface requirements, 28
IOSYS, 32
learning behavior, 39
learning behavior function, 33
learning curve, 46
learning input-output systems, 33
learning time, 39
license, 4
logical design phase, 49
logical space, 41
MAS, 24
mathematical actor story, 24
mathematical AS, 25
mathematical mode, 20
measure behavior, 31
mental ontology, 22
METHOD, 35
model, 32
model as hypothesis, 40
Moran, 34
names, 25
Newell, 34
no 1-1 mapping, 43
non-deterministic behavior, 34
non-empirically motivated, 33
non-functional requirements, 21
objective behavior data, 46
observe behavior, 31
OPERATOR, 35
optimal solution, 20
optimal value, 45
OSC, 17
PARC, 34
PAS, 23
Philosopher of Science, 16
physical space, 42
pictorial actor story, 23
pictorial language, 22
pictorial mode, 20
predict behavior, 31
problem, 20
problem document, 19
process input, 31
properties, 26
questionnaire for feelings, 46
real interface, 32
recruiting executing actors, 42
respond, 31
RIntf, 32
SAS, 27
SELECTION, 35
semantic gap, 19
sense, 31
SEP, 16
SEP Manager, 16
SEP structure, 16
simulated actor story, 27
simulating automaton, 23
simulation mode, 20
stakeholder, 19
story, 20
| TAR, 28, 43 | test series, 46 | usability test, 45 |
| TAS, 23 | textual actor story, 23 | validation phase, 49 |
| task, 19 | textual mode, 20 | viewpoint, 15 |
| task induced actor requirements, 28 | theoretical behavior function, 31 |  |