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

ACTOR ACTOR INTERACTION [AAI] WITHIN SYSTEMS ENGINEERING (SE)
AN ACTOR CENTERED APPROACH TO PROBLEM SOLVING IN ENGINEERING
COMBINING ENGINEERING AND PHILOSOPHY

VERSION 9. AUGUST 2018

UFFMM.ORG
## Contents

*Preface* 11

1 *Introduction* 13

2 *Actor-Actor Interaction Analysis* 17

3 *Actor Story (AS)* 19

4 *AS as Text, Comic, Graph* 23

5 *Actor Model (AM)* 33

6 *Simulation* 45

7 *Algorithmic Verification* 49

8 *Physical Design* 51

9 *Usability Testing* 53

10 *AS-AM Philosophy* 57

11 *Looking Forward* 63
# List of Figures

1.1 Engineering process with different kinds of actors 14  
3.1 Translating the individual world views into three different languages 20  
4.1 Informal example of a word-to-picture lexicon as starting point for a mapping from TAS into PAS 25  
4.2 Simple pictorial actor story associated with the ‘open door TAS’ 25  
4.3 Example of a picture-to-math mapping for properties and objects 27  
4.4 Change event between two states 28  
4.5 Change event with an embedded actor 30  
5.1 Basic Typology of Input-Output Systems 36  
5.2 Electronic door example - bare graph, only nodes 41  
5.3 aai-example electronic door: nodes and minimally labeled edges 41  
5.4 aai example electronic door with nodes, edge-labels, and properties 41  
5.5 aai example with a complete graph (only the edge labels are shortened) 41  
5.6 Graph with complete start state folowed by difference states based on labelled edges 42  
6.1 Creation of the symbolic space by AAI-experts 45  
10.1 The actor story (AS) and the actor models (AMs) as symbolic representations constituting a symbolic space 58  
10.2 Creation of the symbolic space by AAI-experts 59  
10.3 The symbolic space extended with simulators translated into the physical space with physical actors as well as physical assistive actors 60  
10.4 The intended actors of an actor story (AS) are living as real actors usually in more than one actor story 61  
11.1 Vision of a solution for the problem of a stakeholder cooperating with common models in a productive way 65
Dedicated to those who gave us the prior experience and the inspiring ideas to be able to develop the view offered in this book.
Preface

AAI, SE, AI, Philosophy  This book is our first trial to bring together such diverse topics as 'Human-Machine Interaction (HMI), Systems Engineering (SE), Artificial Intelligence (AI), and Philosophy of Science (PoS) in one coherent framework called Actor-Actor Interaction within systems Engineering (AAI-SE).

Overview of the book  The book starts with an introduction presenting all key ideas and how they will form, step by step, a big picture. Then you can dig into each of the topics with more details and with more examples, commented by historical backgrounds and actual discussions in the community. At the end of the book you will find first case studies illustrating how the new framework can be applied to real-world problems. With the final index of key terms you will be able to find the passages in the book where these terms are used.

About the web site  After the publication of this book the accompanying website https://www.uffmm.org/ of the book will offer additional material for the community.

Acknowledgements  This book 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\(^1\), many intensive discussions between Doeben-Henisch and Erasmus about Systems engineering since 1999, a paper written by Doeben-Henisch and Wagner 2007\(^2\) with ongoing discussions since then, a lecture by Doeben-Henisch about formal specification and verification in 2010\(^3\), two papers by Erasmus and Doeben Henisch in 2011\(^4\), more than 22 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 - 2018, 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 discussions between Doeben-Henisch and Idrissi about AI and AAI since 2015.

---

1 He died 10. May 2018.
1

Introduction

The term ‘Actor-Actor Interaction (AAI)’ as used in the title of the book is not yet very common. Better known is the term ‘HMI’ (Human-Machine Interaction) which again points back to the term ‘HCI’ (Human-Computer Interaction). 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.¹

One can observe an explosion of new applications and usages of computer. This caused a continuous challenge of how human persons can interact with this new technology which 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 a computerized device with sensors and actuators. At the same time the collaborators of human persons altered to ‘intelligent’ machines, robots, and smart interfaces. Thus to speak of a ‘human user’ interacting with a ‘technical interface’ seems no longer to be 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 today biological systems (man as well as animals) and non-biological systems. Therefor we decided to talk instead of Human-Machine Interaction (HMI) now of ‘Actor-Actor Interaction (AAI)’.

The term ‘Systems Engineering (SE)’ is well known in the area of engineering,² but not necessarily in connection with the new Actor-Actor-Interaction paradigm. Our motivation to combine the AAI-view with the Systems Engineering view was stimulated by the question whether there exists a framework for AAI analysis which provides all the parameters which an AAI analysis needs.

In systems engineering (cf. figure 1.1) it is common to assume an expert as part of a system engineering process who takes a problem description $D_p$ from a stakeholder, and does some analysis work to find an optimal solution candidate for the problem. Content of this analysis is the task which

¹ For a first introduction see the two human-computer interaction handbooks from 2003 and 2008, and here especially the first chapters dealing explicitly with the history of HCI (cf. Richard W. Pew (2003) , which is citing several papers and books with additional historical investigations (cf. p.2), and Jonathan Grudin (2008) . Another source is the ‘HCI Bibliography: Human-Computer Interaction Resources’ (see: http://www.hcibi.org/), which has a rich historical section too (see: http://www.hcibi.org/hci-sites/history).

² For a first introduction cf. INCOSE (2015)
has to be solved as well as the different kinds of actors, which are involved in this task. Therefore the term Actor-Actor Interaction analysis.

One level above the expert doing the analysis we have the manager of the systems engineering process, who is setting the framework for the process and who has to monitor its working.

Another upper level is the philosopher of science 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) analysis is the subject matter of the expert doing the AAI-analysis work.

For the actor-actor interaction (AAI) analysis as part of a systems engineering process (SEP) the following highly idealized structure is assumed.\(^3\)

\[
AAIA(x) \quad \text{iff} \quad x = \langle A, D, D_P, D_{\text{AAR}}, M_{\text{SR}}, M_\sigma, M_\nu, M_{\text{DIntf}}, M^*_D_{\text{Intf}}, \delta \rangle
\]

\[
\begin{align*}
A & : \text{Set of actors} \\
D & : \text{Set of documents} \\
D_P & : \text{Set of problem documents} \\
M_{\text{SR}} & : \text{Set of behavior models} \\
M_\sigma & : \text{Set of simulator models} \\
M_\nu & : \text{Set of algorithmic verification models} \\
M_{\text{DIntf}} & : \text{Set of real interfaces} \\
M^*_D_{\text{Intf}} & : \text{Set of optimized real interfaces} \\
\delta & = \alpha \otimes \beta \otimes \sigma \otimes \pi \otimes \gamma \otimes \circ \\
\alpha & : A \times D \rightarrow D_P \\
\beta & : A \times D \times D_P \rightarrow M_{\text{SR}}
\end{align*}
\]
This description hides many details but provides enough information to locate the AAI analysis within a systems engineering process.

Thus an actor-actor interaction analysis assumes a set of actors $A$ (stakeholders, experts, ...) and some knowledge represented in documents $D$ which then will be mapped by a process called $\alpha$ into a problem document $D_P$ which contains besides different informations non-functional requirements too. As language used for the generation of the problem document an everyday language $L_0$ is assumed.

Again, actors, knowledge documents as well as the problem document will then be mapped with a process called $\beta$ into a behavior model $M_{SR}$. A behavior model will include an actor story (AS) as well as (optionally) many actor models (AMs). The actor story represents all necessary functional requirements (FR) of the problem and it can include a set of non-functional requirements (NFR) distributed throughout the whole actor story. Thus we have $M_{SR} = AS \cup AM$. The actor story will be presented in multiple modes. First in a textual mode written in some everyday language $L_0$. This textual mode will then be translated into two different modes: in a mathematical mode with language $L_\epsilon$ and into a pictorial mode with a pictorial language $L_{pict}$. The pictorial mode can be used as an artificial model of meaning for the mathematical mode. One needs some mapping (used as a 'lexicon') between an actor story $AS_{pict}$ in pictorial mode and an actor story $AS_\epsilon$ in mathematical mode.

Based on the mathematical mode of an actor story $AS_\epsilon$ one can convert the actor story $AS_\epsilon$ with an algorithm into an automaton $M_\alpha$ which can be run on an appropriate computer as a simulation. The combination of this automaton $M_\alpha$ with an appropriate computer we call a simulator model $M_\sigma$. The whole process preparing a behavior model $M_{SR}$ as a simulator model is called $\sigma$.

Another helpful process is the process named $\nu$. It translates a behavior model $M_{SR}$ with the aid of a temporal logic language $L_{TL}$ and an appropriate algorithm $\alpha$ into a algorithmic verification model $M_\nu$, which can compute the occurrence or non-occurrence of a certain property in the space of possible states of the behavior model. This capability of deciding the occurrence or non-occurrence of certain properties is especially helpful in the case of non-functional requirements. Because non-functional requirements are usually defined by decidable properties attached in a distributed manner to a behavior model such a automatic verification process can check exactly these distributed properties.

To test the usability of the behavior model one has to translate the logical concept of the assistive actors serving as interfaces into a physical appearance of the assistive actors and during this translation in a process called $\gamma$ one has to use knowledge from the actor-actor induced requirements (AAR) as well as knowledge from Psychology to design a physical appearance of

\[
\begin{align*}
\sigma &: A \times D \times M_{SR} \mapsto M_\sigma \\
\nu &: A \times D \times M_{SR} \mapsto M_\nu \\
\gamma &: A \times D \times M_{SR} \times D_{AAR} \mapsto M_{DInt} \\
\omega &: A \times D \times M_{SR} \times D_{AAR} \times M_{DInt} \mapsto M_{DInt}^* 
\end{align*}
\]
the assistive actors $M_{DInt}$ which can be tested by real users functioning as executive actors.

Finally, to get real data from real users for a usability test one has to arrange an experimental setting whereby a real user – corresponding to the assumed AAR profiles – is challenged to do the required task(s) of the problem. This behavior is kept in a protocol. After this objective part of the test the user is invited for a small questionnaire to write down his judgments regarding his feelings during the test as well as the circumstances of his feelings. Observation protocols and questionnaires of a set of $n$ test-persons ($n = \{5 - 9\}$) will then be evaluated. After this evaluation the developer team can consider some possible improvements, and – if improvements have been realized – the tests can be repeated with new test-persons. This whole procedure of (testing - improvements) can be repeated several times; at least three times. How many repetitions are finally ‘optimal’ is actually an open question. It depends to a high degree from the parameter measuring the learning capacity of the test persons. How often should one test a test-person and in which timely distance between each test? The whole evaluation process with all possible repetitions is called the $\omega$-process.

Philosophy of the AAI-Expert The ‘Philosophy of the AAI-Expert’ is centering around the findings of modern Biology and Psychology. Its aim is to explain why a human expert is able to use a formal language, here the set theoretical language $L$, to talk about his experiences of the empirical world. What Biology and Psychology are telling us is that the communication of the experts is grounded in their cognitive machinery embedded in their brains. Because the human brain in the body is not directly interacting with the outside world but mediated by sensors and actuators the brain 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. For more details see chapter 10 ‘AS and AM Philosophy’.

One conclusion from these considerations is that the reality for a human person is basically given as a stream of neural events, partially translated into phenomena of the consciousness, 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.

This, to understand how an AAI-expert perceives his world, generates internal models, and how he is communicating with others, this is the subject for a philosophical grounding of the following AAI analysis theory.
2 Actor-Actor Interaction Analysis

In the following text we describe the actor-actor interaction analysis – short: AAI analysis – by following the schema 1.1 from the introduction. On account of the inherent complexity of some of these themes we dedicate for these complex topics complete chapters.

Problem Document

According to the schema 1.1 the first sub-process is given by the process \( \alpha : A \times D \mapsto D_P \). This process generates a problem document \( D_P \). This is the result of a communication process between some stakeholders (SH) and some experts (EXP) who represent different kinds of actors \( A \). The original problem \( P \), which a stakeholder wants to be solved, is assumed to be described in some introductory document \( D \).

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.\(^1\)

Additionally to the problem described in the problem document \( D_P \) a finite set of special constraints (C) can be given in this document too, which correspond to the traditional 'non-functional requirements (NFR)'. Non-functional requirements are those which describe properties of a whole process, which can not be recognized by an individual, isolated property alone. Examples are 'safety', 'security', 'cost efficiency', 'barrier freeness', 'competitive with regard to a certain market', 'reliability', etc. To apply such non-functional requirements one has to define a set of operational criteria which all-together represent a non-functional requirement. This set of operational criteria must be associated with that process – called 'actor story (AS)' (see below) –, which realizes the intended problem. If the criteria are all 'satisfied' then the non-functional requirement is fulfilled, otherwise not.

Check for AAI-Analysis

The problem given in a systems engineering process must not necessarily be appropriate for an AAI analysis. Therefore it makes sense to do some test in advance whether the problem in a problem document \( D_P \) is fitting to

an AAI analysis. Such a test of the problem in a problem document checks for the occurrence of the following properties:

1. Does the problem include at least one task (T) to be realized to reach a solution?

2. Does the problem include an environment (ENV) for the task?

3. Does the problem include at least one executive actor (ExecA) as the intended user, which shall use some technology as an assistive actor (AssisA) – often called interface – to run the task?

   If all three question will be answered affirmatively then the problem can be analyzed within an AAI analysis.

**Behavior Model**

Following the schema 1.1 further we meet the next sub-process called beta \( \beta : A \times D \times D_p \rightarrow M_{SR} \). This process generates a behavior model \( M_{SR} \) which includes all information which is necessary to realize the task(s) necessary for the realization of the intended problem.

Looking deeper into the structure of the behavior model one meets a rather complex conceptual machinery for which we will dedicate individual chapters.

1. The first chapter called *actor story (AS)* describes a process rooted in a series of connected states which together represent – like a story – the necessary situations which have to be run through to reach the characterized goal states of the process. The actor story represents all necessary functional requirements (FR) of the problem and it can include a set of non-functional requirements (NFR) distributed throughout the whole actor story. The actor story will be presented in multiple modes. First in a *textual* mode written in some everyday language \( L_0 \). This textual mode will then be translated into two different modes: in a *mathematical* mode with language \( L_e \) and into a *pictorial* mode with a pictorial language \( L_{pict} \). The pictorial mode can be used as an artificial model of meaning for the mathematical mode. One needs some mapping (used as a 'lexicon') between an actor story \( AS_{pict} \) in pictorial mode and an actor story \( AS_e \) in mathematical mode.

2. The second chapter describes – optionally – *actor models (AM).* These are models of behavior of actors which are part of the actor story. An actor model characterizes the overt behavior of an actor by the construction of an explicit behavior function rooted in the internal states (IS) of an actor. The concept of the actor model allows the introduction of the topic of artificial intelligence (AI) dealing with that subset of actor models which represent intelligent behavior as well as learning behavior. 'Intelligence' and 'learning' are two independent properties!
3

Actor Story (AS)

First Concepts

HOW IT STARTS: As described in the chapter 2 'AAI analysis' the starting point for an AAI analysis is a problem document $D_P$ which describes in a first way which kind of a problem a stakeholder wants to be solved. As identifying criteria whether the problem at hand is appropriate for an AAI analysis are mentioned the existence of at least one task, an associated environment, at least one assistive and executive actor and optionally some non-functional requirements.

Following this setting this chapter is dedicated to the construction of a behavior model $M_{SR}$ which is assumed to consist of an actor story (AS) as well as – optionally – actor models (AMs). Actor models are discussed in the next chapter.

ACTOR STORY CONCEPT: The concept of the actor story (AS) is inspired by the fact that every task can be understood as a sequence of situations – here also called states – which are connected by events which cause some change in a given state. It is assumed that there is at least one start state and at least one goal state which represents the ‘solution’ of the task.

A state representing some situation is understood as a set of properties, which either can be verified in a real given situation or which are candidates to be able to become verified in a real situation. The last case is typical for properties in a state sequence which describes a possible sequence of some conceivable future. State descriptions which are considered as impossible to be become decidable in a real situation are not accepted as possible states.

Subsets of properties in a state can be understood as objects. Special kinds of objects are actors which are input-output systems (IOSYS) which can perceive some properties of the state they are in as possible inputs (I) as well they are able to produce some output (O) which can be an event which causes some change with regard to the properties of the state.

DOMAIN OF REFERENCE

All these mentioned terms like 'task', 'state', 'event', 'actor' and the like constitute what usually is called the domain of reference $D_{Ref}$. These are all the internal states a brain assumes as something present in the
consciousness of a person or which can become conscious by remembering or thinking. Another common way of speaking in this context is to speak of mental models, which are present in our thinking and which we use to understand the world.

3rd Person View: 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 its 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. An inner state corresponds to what often is called the 1st-person view. To overcome the limits of a 3rd-person view one has to construct additional models called Actor Models (AMs) as described in the next chapter.

Functional and Non-functional Requirements: The actor story represents all necessary functional requirements (FR). For non-functional requirement (NFR) see chapter 7 ‘Algorithmic Verification’.

Coordination by Communication

Whatever internal states an individual brain may possess, if the brain wants to coordinate its contents with other brains it has to enable some kind of communication which allows a continuous encoding of these internal states into the expressions of some language $L$ and simultaneously a decoding of these expressions into the internal states of another brain.

For such communications the homo sapiens population has developed since many thousand years primary, everyday languages $L_0$ which use primarily sounds $L_{0,snd}$ (enhanced by mimics, gestures, etc.), which later have been mapped into written symbols $L_{0,symb}$ too. Thus there exists an internal mapping between both modes: $sndSymb : L_{0,snd} \longrightarrow L_{0,symb}$ and $symbSnd : L_{0,symb} \longrightarrow L_{0,snd}$. These mappings are not really 1-to-1, because the utterance situation can not completely be resolved in the stream of sounds and even less into a stream of written symbols. Therefore is a re-mapping from the written symbols into the utterance situation only possible in a reducing way; especially it is difficult or even impossible to
re-map written symbols clearly and completely back into the internal states of the source of the communication.

From these restricted re-mappings it follows that an everyday-language communication includes always some fuzziness which urges the participants to re-check as often as possible whether their understanding based on these communications is in a sufficient agreement.

Dealing with this ‘natural fuzziness’ from the point of engineering it became clear that it could be helpful to use different types of languages – in this text called different modes – which show different ways of encoding-decoding. As you can see in the figure 3.1 three modes are favored: the usual everyday language mode $L_{0,symb}$ as basic or standard mode; a pictorial mode $L_{pict}$ as an additional mode, as well a mathematical mode $L_{math}$. From the point of encoding-decoding it is possible to use the pictorial and the mathematical mode as ‘complementary to each other’: the pictorial mode describes the ‘domain of reference’ for the mathematical mode. To use the pictorial mode as an artificial model of meaning for the mathematical mode One needs some mapping (e.g. in a simple version as a ‘lexicon’) between an actor story $AS_{pict}$ in pictorial mode and an actor story $AS_{e}$ in mathematical mode. In case of the everyday language the domain of reference is only given in the inner states of the brain, which to some extend can be correlated with properties of the body world. In case of the mathematical mode there are three languages combined: (i) a language of graphs $L_{math,graph}$, a language of properties $L_{math,prop}$, and a language of changes $L_{math,x}$.

The ‘mathematical language’ $L_{math}$ can easily be used for logical proofs, for automated computations, as well as for computer simulations. The language of mathematical graphs additional enriched with formal expressions for properties and changes between states allows an automatic conversion into automata which can simulate all these processes. Additionally one can apply automatic verification for selected properties, e.g. for non-functional requirements.

The additional actor models described after the actor story are a special extension of the actor story and have to be included in the simulation mode.

---

1 Later in this book it will be shown that the mathematical language $L_{math}$ can be enhanced by even more languages.

2 For simulation see chapter 6.

3 For automatic verification see chapter 7.

4 For actor models see chapter 5.
4

AS as Text, Comic, Graph

This chapter describes in more detail the three main modes of an actor story.

Textual Actor Story (TAS)

An actor story AS in the textual mode – here called ‘textual actor story (TAS)’ – is a text composed by expressions of some everyday language \( L_0 \) – default here is English \( L_{EN} \). Such a text describes as its 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.

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 \( t \) \((t \in T)\).

A task is assumed to be a sequence of states with a start state \( s_{start} \) and at least one 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 the goal state. The states between the start and the goal state are called intermediate states. For a feasible task there should exist at least one path from a start state to a goal state.\(^1\)

Locating the domain of reference \( D_{Ref} \) inside the brain in close association with the consciousness one can postulate a basic mapping for the textual actor story as follows:

\[
\text{textual} : D_{Ref} \longrightarrow TAS
\]

Because the domain of reference is ‘inside’ an expert it can not directly be shown. If two experts want to compare whether a given textual actor story \( t \) is ‘appropriate’ compared to the presupposed meaning \( m \) then this is only possible if the meaning \( m \) can be correlated with properties ‘outside

\(^1\) 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 brain’ like in the case of ‘empirical matters’ which occur ‘between’ the experts and which can be ‘empirically observed’. If the encoding of these ‘empirical observations’ into some text is sufficiently similar between the both experts then they will usually ‘agree’ and confirm that this text ‘represents’ the intended meaning m in an ‘equally appropriate way’. In this sense is the development of a ‘common meaning’ between different experts an adaptive process using lots of ‘working hypotheses’ where the ‘right ones’ will be filtered out by ‘confirmation’ and ‘dis-confirmation’. As one can infer from these varying processes there is space enough for errors and misunderstanding. This always highly indeterministic situation in the context of the encoding of meaning is often called semantic gap.

TAS-Example: Open Door

• +++++ Begin of example ++++
• Name: Open Door
• Start: A person A stands before a closed door. Besides the door there is a numeric keypad.
• Entering Key: The person A enters a key NNN into the keypad. GOTO state named ‘Goal’. ELSE GOTO state named ‘Start’.
• Goal: The door is open.
• +++++ End of example ++++

Pictorial Actor Story (PAS)

The notorious semantic gap problem associated with the encoding of meaning into a textual actor story can be reduced to some extend by using a pictorial actor story as well. Mapping the intended meaning m, which is already encoded in a textual actor story, furthermore in a pictorial actor story, which has sufficient resemblances with the intended meaning, then it is possible to show parts of the encoded meaning m more directly.

This approach presupposes that for a standard user the world of objects inside the brain 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 actor story can substitute the intended meaning m to some degree if the pictorial language provides pictures which are structurally sufficient similar to the perceived visual structure of the observer.

To preserve a consistency between the textual and the pictorial mode one needs a mapping between the domain of reference together with a textual actor story into a pictorial actor story:

\[
pictorial : D^{\text{Ref}} \times TAS \leftrightarrow PAS
\] (4.2)

Because the intended meaning m which shall be encoded is – like in the case of the textual actor story – partially rooted in the inner states of
the experts one can represent only the textual and the pictorial part of the mapping directly. Nevertheless if the pictorial part of the encoding has indeed sufficient similarities with the presupposed intended meaning then the pictorial part can enhance the understanding between different experts and can thereby reduce possible misunderstandings.²

² It is desirable to start a series of psychological experiments comparing the degree of understanding using (i) only textual mode, (ii) only pictorial mode, as well as (iii) a combination of textual and pictorial mode in parallel.

Figure 4.1: Informal example of a word-to-picture lexicon as starting point for a mapping from TAS into PAS

PAS Example: Open Door

As the informal figure 4.1 shows one can structure the mapping from TAS to PAS as follows:

\[
\begin{align*}
\text{state} & \leftrightarrow \text{single picture frame} \quad (4.3) \\
\text{object } \cup \text{ properties} & \leftrightarrow \text{shape } \cup \text{ text} \quad (4.4) \\
\text{relation} & \leftrightarrow \text{shapes in neighbourhood } \cup \text{ text} \quad (4.5) \\
\text{action} & \leftrightarrow \text{state and followup states} \quad (4.6)
\end{align*}
\]

For a full specified mapping one needs much more details regarding e.g. different sizes and complex arranged properties. Nevertheless one can infer here the basic structure. A possible realization of an pictorial actor story referring to the TAS 'Open Door' (cf. TAS-example 4.1.1) could be the following one:

Figure 4.2: Simple pictorial actor story associated with the ‘open door TAS’
Mathematical Actor Story (MAS)

The construction of a mathematical actor story (MAS) has similar presuppositions as the construction of a pictorial actor story: it is assumed that for a standard user the world of objects inside the brain 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 mathematical actor story can map the intended meaning $m$ to some degree if the mathematical language provides structures which are sufficient similar to the conscious structures of the observer.

To preserve a consistency between the textual and the mathematical mode one needs a mapping between the domain of reference together with a textual actor story into a mathematical actor story:

$$math : D^{Ref} \times TAS \leftrightarrow MAS$$ (4.7)

The following table indicates how one can structure the mapping from $D^{Ref} \times TAS$ to MAS:

<table>
<thead>
<tr>
<th>State</th>
<th>Vertex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object $\cup$ Properties</td>
<td>Set of properties$^1$ with subsets</td>
</tr>
<tr>
<td>Relation</td>
<td>Properties$^n$</td>
</tr>
<tr>
<td>Action</td>
<td>Vertex and followup vertices with change $-$ expressions</td>
</tr>
</tbody>
</table>

This mapping can be interpreted as follows:

1. **state $\leftrightarrow$ vertex**: A state or situation given as a spatial structure corresponds in a graph $\gamma$ to a vertex (also called 'node') $v$.

2. **object $\cup$ properties $\leftrightarrow$ set of properties$^1$ with subsets**: The basic units of a state as vertex are 1-ary property-expressions of a property language $L_{prop}$. Certain subsets of these properties can be defined as objects, usually assumed to be actors. Actors are assumed to be input-output-systems. Therefore there exists a mapping from vertices in properties like: $\lambda : V \mapsto 2^{L_{prop}}$.

3. **relation $\leftrightarrow$ properties$^n$**: There are also special properties called n-ary relations which hold between objects.

4. **action $\leftrightarrow$ vertex and followup vertices with change $-$ expressions**: If at least one property changes in a vertex $v$ on account of some action triggered by an object in this vertex then one has to generate a successor vertex $v'$ with a change-expression of some change language $L_x$ attached to the edge $(v,v')$. For these change expressions one needs an additional mapping like $\epsilon : V \times V \mapsto 2^{L_x}$.

A graph $\gamma$ extended with properties and changes is called an extended graph $\gamma^+$.
Thus we have the extended graph $\gamma^+$ given as:

$$\gamma^+(g) \iff g = (V, E, L_{\text{prop}}, L_x, \lambda, \epsilon) \quad (4.12)$$

$$E \subseteq V \times V \quad (4.13)$$

$$\lambda : V \rightarrow 2^{L_{\text{prop}}} \quad (4.14)$$

$$\epsilon : E \rightarrow 2^{L_x} \quad (4.15)$$

Figure 4.3 shows a simple example of a picture-to-math mapping for objects and properties. Because it is assumed here that the pictures represent sufficiently well the intended meaning of the domain of reference one can use the pictures as point of reference for the intended meaning space.

From this figure one can infer that properties can be grouped as objects. Objects can be mapped into names like {'D1', 'U1', 'Kp'} , 1-ary properties in 1-ary property expressions like {'OPEN()', 'USER()', 'KEYPAD()'} and 2-ary properties (also called relations) are mapped into 2-ary property expressions like 'BEFORE(,)'.

Combining property expressions with names to property statements like {'OPEN(D1)', 'USER(U1)', 'KEYPAD(Kp)', 'BEFORE(U1, Kp)'} allows statements about objects and their properties. Thus we have:

1. 'OPEN(D1) := 'The door with name 'D1' is open’
2. 'USER(U1) := 'The object with name U1 is a user’
3. 'KEYPAD(Kp) := 'The object with name Kp is a Keypad’
4. 'BEFORE(U1, Kp) := 'The object with name U1 is before the object with name D1’
Because a change is related to the change of at least one property in one state \( v' \) compared to the (in time) preceding state \( v \) one has to have expressions which can ‘talk’ about consecutive states and the related property changes. The nucleus of a change-statement is the name of an action understood as an operation mapping a set of properties into another set of properties like:

\[
\text{action} : \text{state} \rightarrow \text{state} \tag{4.16}
\]

\[
\text{action} : 2^{L_{\text{prop}}} \rightarrow 2^{L_{\text{prop}}} \tag{4.17}
\]

A practical device to realize such a mapping is the following procedure:

1. Given a vertex \( v \) with the properties \( \{P_1, ..., P_n\} \)
2. Given an action with the name ‘action’
3. Construct a new vertex \( v' \) as follows: (i) copy vertex \( v \) to vertex \( v' \); (ii) delete all properties in \( v' \) which shall be cleared out written as \( d\{...\} \); (iii) create new properties which shall be introduced, written as \( c\{...\} \). The deletion-creation operations are processed in a certain order: first deletion and then addition, \( \text{change} = d \otimes c \).
4. The expression ‘action() : d{}, c{}’ defines the meaning of the operation ‘action’ by the deletion and creation information.
5. The full change-expression is written as follows: \( \langle v, v', \text{action} - \text{name()} : d\{\}, c\{} \rangle \).

An example is given in figure 4.4: The occurrence of a change is represented by two vertices \( v, v' \) connected by an edge \( c \) as \( v \rightarrow v' \). 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 is represented in a formal change-expression containing a list of properties to be deleted as \( d\{p_1, p_3, ...\} \) and properties to be newly created as \( c\{p_2, p_4, ...\} \).
These conventions define the actor story as formal mathematical graph enhanced by formulas to represent properties and formal expressions for changes.

**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, \( m_{pm} : \text{PAS} \rightarrow \text{MAS} \). The figure 4.3 gives a fist idea how such a mapping can be constructed. More details have o follow ...

**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 \( RProf_A \). 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 \( A_{assist} \). 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 \( RProf_{requ} \) 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, \( \sigma : RProf_{requ} \times RProf_{avail} \rightarrow [0, 1] \).

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 \( \chi : A_{avail,exec} \times RProf_{requ} \rightarrow A_{requ,exec} \). 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 $\text{Int} f_{A,\text{exec}}$ and similarly the sum of all possible perceptions and reactions of the assistive actor the interface of the assistive actor $\text{Int} f_{A,\text{assis}}$, 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 cannot 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 4.5.

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 $\langle v, v', \text{press}(\text{BUTTON}(B1)), d(\text{not} \text{- pressed}(B1)), C : (\text{pressed}(B1))\rangle$. 

![Figure 4.5: Change event with an embedded actor](image-url)
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*:

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

(5.1)

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:

\[
\phi : I \longrightarrow O
\]

(5.2)

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*.

\[
\phi_e \subseteq \phi
\]

(5.3)
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* \(I\) for the receiving actor as well as (ii) *outputs* \(O\) from the actor 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 \(E\).

To define this more explicitly we will define the following terms: *Environment* \((E)\), *Input-Output system* (IOSYS) as well as *Actor* \((A)\). As Interface between the actor and the environment we have also the *Basic Interface* (BIntf).

The actors \((ACT)\) are understood as input-output systems (IOSYS).

It is difficult to describe formally the interaction between an environment \((E)\) and an actor \((A)\). The environment offers existing properties which can change from time to time. The possible ‘effect’ of these properties and their changes depend on the built-in sensor functions of the actor. Thus the stimulus-function \(\sigma\) of the environment can map some subset of properties of the environment onto some actor, but which effect these mapped properties will have as internal input \((I)\) in the actor depends from the actor-specific sensor functions \(\sigma_A\). Thus we have a chain \(\sigma_E : 2^\Pi \mapsto ACT\) and then \(\sigma_A : \text{rn}(\sigma_E) \mapsto I_A\). The same is true for the backward chain from the outputs of an actor to the environment: An actor \(A\) has generated internally some outputs \(O_A\) which are first translated by its motor function \(\mu_A\) into some external properties of the actor \(A\), which in turn are then translated by the response function of the environment \(\mu\) into some effects represented as deletion of existing properties \(2^\Pi^-\) as well as of creation of new properties \(2^\Pi^+\): \(\mu_A : O_A \mapsto O_{A,\text{resp}}\) and then \(\mu : \text{rn}(\mu_A) \mapsto 2^\Pi^- \cup 2^\Pi^+\).

Thus we get a *hierarchical embedding* of structures:

\[
\begin{align*}
\text{ENV}(E) & := \text{Environment } E \\
\text{ENV}(E) & \text{ iff } E = (\Pi, ACT, \sigma, \mu) \\
\Pi & := \text{Set of properties} \\
ACT & := \text{Set of actors} \\
ACT & \subseteq 2^\Pi \\
\sigma & : 2^\Pi \mapsto ACT(\text{stimulus function}) \\
\mu & : O_{ACT,\text{resp}} \mapsto 2^\Pi (\text{response function})
\end{align*}
\]

and:
\[ ACT(A) := \text{Actor } A \quad (5.5) \]
\[ ACT(A) \text{ iff } A \in ACT \land A = \langle I, O, IS, \sigma, \mu \rangle \]
\[ I_A := \text{Input} \]
\[ O_A := \text{Output} \]
\[ \sigma_A : \text{rn}(\sigma_E) \rightarrow I_A \]
\[ \mu_A : O_A \rightarrow O_{A,resp} \]

and:
\[ IOSYS(S) := \text{Input – Output System} \quad (5.6) \]
\[ IOSYS(S) \text{ iff } S = \langle I, O, IS, \phi \rangle \]
\[ I := \text{Input} \]
\[ O := \text{Output} \]
\[ IS := \text{Internal States (can be empty)} \]
\[ \phi : I \times 2^{\Pi} \times 2^{\Pi} \times O \]

An input-output system (IOSYS) can be defined independent from sensor and motor functions but then the actor is 'disconnected' from every kind of environment. Thus we use the term 'input-output system' if we talk about actors in a more abstract way and we use the term 'actor' for actors if we talk about actors as input-output systems somehow embedded in some environment.¹

With these clarifications it becomes clear that the basic interface (BIntf) of an actor A in the environment E has not to be defined with the 'internal' inputs and outputs of an actor but by the image/range of the environment-stimulus function \( \text{rn}(\sigma_E) \) as well as the response-values of the actor \( O_{A,resp} \). Thus we have:
\[ BIntf_{A,E} = \{ x | x \in \text{rn}(\sigma_E) \times O_{A,resp} \} \]

This definition shows not only (as stated above) that the basic interface is a finite set of input-output pairs, but additionally the observed inputs are mere estimates of inputs because the observed stimuli from point of view of the environment are not necessarily the inputs inside of the actor. The stimulus function of the actor in connection with the internal states usually does modify the outside-stimuli in specific ways.

**Real Interface (RIntf)** The basic interface (BIntf) as logical concept 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

¹ Here is the environment defined by the actor story.
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’.

**Input-Output Systems Basic Typology**

![Diagram of Input-Output Systems Basic Typology](image)

With the basic parameters Input (I), Output (O) as well as Internal States (IS) one can derive some basic typology of input-output systems(cf. figure 5.1).

A first case is the random case where the output of a system will be completely random within the space of possible outputs independent of the input. Thus with regard to the set of random possible system-dependent outputs $O_{Random, SYS}$ every output can occur.
A second case is the fixed (deterministic) case where a subset of the system-dependent outputs $O_{\text{Fixed},\text{SYS}}$ is in a static manner associated with a certain input. This determination of a certain subset of the system-dependent outputs represents some sort of a bias; not the whole set is possible, but only a pre-defined subset.

The final case describes an incrementally fixed case where the system can change its behavior during runtime $O_{\text{Sel},\text{SYS}}$ depending on some kinds of rewards which can be part either of the external input $I$ or of some internal states $I_{\text{REW}}$. Although the set of system-dependent outputs can change, the set of possible outputs represents a certain subset of all the possible outputs and therefore is nevertheless by this selection a bias which is influenced by the rewards.

If one steps back even more and takes a look to the three types $O_{\text{Random},\text{SYS}}, O_{\text{Fixed},\text{SYS}}, O_{\text{Sel},\text{SYS}}$ then one can compare these special sets with the general set of system-dependent outputs $O_{\text{SYS}}$ and the set of possible outputs offered by the actor story as the world (W) given as $O_{\text{W}}$. If one takes the possible outputs of the world called $O_{\text{W}}$ as point of reference then the system dependent outputs $O_{\text{Random},\text{SYS}}, O_{\text{Fixed},\text{SYS}}, O_{\text{Sel},\text{SYS}}, O_{\text{SYS}}$ are usually true subsets of the possible world outputs and there can be intriguing overlaps between $O_{\text{Random},\text{SYS}}, O_{\text{Fixed},\text{SYS}}, O_{\text{Sel},\text{SYS}}$. There can be cases that the learning system with its output set $O_{\text{Sel},\text{SYS}}$ is weaker then the system with a fixed output set $O_{\text{Fixed},\text{SYS}}$ and this in turn can be weaker than a random system with the random output set $O_{\text{Random},\text{SYS}}$. Whether this is the case or not depends from many parameters and has empirically to be checked by appropriate tests.

Learning Input-Output Systems From this it follows that the ‘basic interface (Blntl)’ is usually only a subset of the behavior function of a learning system. This means for 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.

Another point is related to the possible reward parts of the external inputs and/or the internal states of an actor. Because learning depends radically on these ‘rewards’ to receive some ‘bias’ to be able to ‘select’ an appropriate subset of possible behavior within a world one has to study these rewards within the logic and dynamics of the actor. The main question is under which conditions a system can approach the optimal output-space using rewards. This assumes that it is possible to determine the optimal space somehow, at least by the ‘rewards’. In the physical world with biological systems the available rewards are results of some past environments. This does not guarantee success in the future. Therefore the main problem is to find new rewards which are more appropriate to enable success in future environments which are usually not completely known during the time of decision making.
Empirically and Non-Empirically Motivated

The general definition of a learning input-output 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)² one can assume at least three sub-functions within the general behavior function:

$$\phi = \phi_{perc} \otimes \phi_{cogn} \otimes \phi_{mot}$$

(5.7)

$$\phi_{perc} := Perception$$

(5.8)

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

(5.9)

$$VB := Visual buffer$$

(5.10)

$$AB := Auditory buffer$$

(5.11)

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

(5.12)

$$\phi_{cogn2} : M_{STM} \times M_{LTM} \rightarrow M_{STM} \times M_{LTM}$$

(5.13)

$$\phi_{cogn1+2} := Cognition$$

(5.14)

$$\phi_{mot} : M_{LTM} \rightarrow O$$

(5.15)

$$\phi_{mot} := Motor activity$$

(5.16)

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:

$$VB \cup AB \cup M_{STM} \cup M_{LTM} \subseteq IS$$

(5.17)

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 5.1) 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.\(^3\)

- **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 \(\text{??}\) a rule of a GOMS model has the logical format:

\[
\text{IF } I = X \land IS = Y \quad \text{THEN } \quad IS = Y' \land O = Z
\]  

(5.18)

**Example: An Electronically Locked Door** For the following demonstration we use the simple example of an electronically locked door.\(^4\)

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 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 = \(U1 \cup S1 \cup Env1\)
   
   \(U1 = \{\text{USER}(U1)\}\)
   
   \(S1 = \{\text{SYSTEMINTF}(S1), \text{KEYPAD}(K1), \text{PART-OF}(K1, S1), \text{KEY}(Ka), \text{KEY}(Kb), \text{KEY}(Kc), \text{PART-OF}(\{Ka, Kb, Kc\}, K1)\}\)
Env1 = {DOOR(D1), CLOSED(D1)}

Meaning: 'U1' is the name of a user, 'S1' the name of a system-interface, and 'Env1' is the name of an environment. All three 'U1, S1, C1' are names for subsets of properties of state Start.

2. CHANGE-AS: ⟨Start, Start, push(not(Ka), K1), d(), c()⟩, ⟨Start, Q2, push(Ka, K1), d(), c(PRESSED(Ka))⟩,

3. Q2 = U1 U S1 U Env1
   U1 = {USER(U1)}
   S1 = {SYSTEMINTF(S1), KEYPAD(K1), PART-OF(K1, S1), KEY(Ka), KEY(Kb), KEY(Kc), PART-OF(⟨Ka, Kb, Kc⟩, K1), PRESSED(Ka)}
   Env1 = {DOOR(D1), CLOSED(D1)}

4. CHANGE-AS: ⟨Q2, Start, push(not(Kb), K1), d(), c()⟩, ⟨Q2, Q3, push(Kb, K1), d(), c(PRESSED(Kb))⟩,

5. Q3 = U1 U S1 U Env1
   U1 = {USER(U1)}
   S1 = {SYSTEMINTF(S1), KEYPAD(K1), PART-OF(K1, S1), KEY(Ka), KEY(Kb), KEY(Kc), PART-OF(⟨Ka, Kb, Kc⟩, K1), PRESSED(Kb)}
   Env1 = {DOOR(D1), CLOSED(D1)}

6. CHANGE-AS: ⟨Q3, Start, push(not(Ka), K1), d(), c()⟩, ⟨Q3, Goal, push(Ka, K1), d(CLOSED(D1)), c(PRESSED(Ka), OPEN(D1))⟩

7. Goal = U1 U S1 U Env1
   U1 = {USER(U1)}
   S1 = {SYSTEMINTF(S1), KEYPAD(K1), PART-OF(K1, S1), KEY(Ka), KEY(Kb), KEY(Kc), PART-OF(⟨Ka, Kb, Kc⟩, K1), PRESSED(Ka)}
   Env1 = {DOOR(D1), OPEN(D1)}

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) 5, and Gansner et.al. (2004) 6. For a tutorial see Gansner et.al (2015) 7.

For practical reasons it seems that the last version, figure 5.6, 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.

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

---


Figure 5.2: Electronic door example - bare graph, only nodes

Figure 5.3: aai-example electronic door: nodes and minimally labeled edges

Figure 5.4: aai example electronic door with nodes, shortened edge-labels, and subsets of properties

Figure 5.5: aai example with a complete graph (only the edge labels are shortened)
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 U1
2. INPUT U1 = VB; OUTPUT U1 = MOT
3. GOAL: OPEN(D1) & DOOR(D1)
   (a) IF VB='CLOSED(D1)' & IS='MEM(U1,(Ka,0,Kb,0,Ka,0))' THEN IS='MEM(U1,(Ka,1,Kb,0,Ka,0))' & MOT='push(Ka,K1)'
   (b) IF VB='CLOSED(D1)' & IS='MEM(U1,(Ka,1,Kb,0,Ka,0))' THEN IS='MEM(U1,(Ka,1,Kb,1,Ka,0))' & MOT='push(Kb,K1)'
   (c) IF VB='CLOSED(D1)' & IS='MEM(U1,(Ka,1,Kb,1,Ka,0))' THEN IS='MEM(U1,(Ka,1,Kb,1,Ka,1))' & MOT='push(Ka,K1)'
   (d) IF VB='OPEN(D1)' & IS='MEM(U1,(Ka,1,Kb,1,Ka,1))' THEN IS='MEM(U1,(Ka,0,Kb,0,Ka,0))' & MOT='movesthrough(U1,D1)'

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

   The system interface S1 has its own GOMS-Model.

1. GOMS MODEL FOR SYSTEM-INTERFACE S1
2. INPUT S1 = K1; OUTPUT S1 = states of the door {CLOSED, OPEN}
3. GOAL: OPEN(D1) & DOOR(D1)
   (a) IF K1='KEY-PRESSED(K1,Ka)' & IS='CODE(C2,(Ka,0,Kb,0,Ka,0))' THEN IS='CODE(C2,(Ka,1,Kb,0,Ka,0))'
   (b) IF K1='KEY-PRESSED(K1,not(Ka))' & IS='CODE(C2,(Ka,0,Kb,0,Ka,0))' THEN IS='CODE(C2,(Ka,0,Kb,0,Ka,0))'
   (c) IF K1='KEY-PRESSED(K1,Kb)' & IS='CODE(C2,(Ka,1,Kb,0,Ka,0))' THEN IS='CODE(C2,(Ka,1,Kb,1,Ka,0))'
(d) \( \text{IF } K1 = \text{KEY-PRESSED}(K1, \text{not}(Kb)) \) & \( \text{IS} = \text{CODE}(C2,(Ka,1,Kb,0,Ka,0)) \)
THEN \( \text{IS} = \text{CODE}(C2,(Ka,0,Kb,0,Ka,0)) \)

(e) \( \text{IF } K1 = \text{KEY-PRESSED}(K1, Ka) \) & \( \text{IS} = \text{CODE}(C2,(Ka,1,Kb,1,Ka,0)) \)
THEN \( \text{IS} = \text{CODE}(C2,(Ka,1,Kb,1,Ka,1)) \) & \( \text{OUT} = \text{OPEN}(D1) \)

(f) \( \text{IF } K1 = \text{KEY-PRESSED}(K1, \text{not}(Ka)) \) & \( \text{IS} = \text{CODE}(C2,(Ka,1,Kb,1,Ka,0)) \)
THEN \( \text{IS} = \text{CODE}(C2,(Ka,0,Kb,0,Ka,0)) \)

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

**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)\textsuperscript{8}, Rost (2009)\textsuperscript{9}, Rost (2013)\textsuperscript{10}).

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)\textsuperscript{11} and a famous experiment with Tolman (1948)\textsuperscript{12} 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)\textsuperscript{13}, English translation\textsuperscript{14})

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 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 \(\varphi\) represent nothing else as a


\textsuperscript{12} 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


hypothesis which is given in the format of a formal model. The formal space for such hypothetical models is infinite.
6 Simulation

**General Outline:** Figure 6.1 outlines the symbolic space which is basically constituted by an actor story in mathematical mode \( (AS) \) and – optionally – by several actor models primarily also in mathematical mode \( (AM) \). In both cases one can compile these representations into equivalent algorithmic representations which represent automata \( M \). These automata are still part of the symbolic space. If one feeds these algorithmic versions into an appropriate physical computer, then one has a full functioning simulation model \( M \).

While the simulation model of the actor story \( M_{\text{AS}} \) written as \( M_{\Sigma} \) serves as the overall framework of the simulation – comparable to the world of a computer game – are the different simulation models of the actor models \( M_{\text{AM}} \) written as \( M_{\tau} \) individual actors operating in the framework of the actor story.

**Algorithmic Conversion:** A given extended graph \( \gamma \) can be mapped into an automaton \( M_{\alpha} \) by a direct mapping. As starting point we take an extended ordered graph (EOG) as follows:\(^1\)

\[
\text{EOG}(x) \iff x = (V, I, F, \Pi, \chi, E, \lambda, \chi)
\]

\[
V := \text{finite set of vertices}
\]

\(^1\) For a good introduction to formal languages and automata see e.g. Hopcroft and Ullman (1979)


---

\( \text{EOG}(x) \) : Extended Ordered Graph

\( V \) : Finite set of vertices

\( I \) : Set of inputs

\( F \) : Set of final states

\( \Pi \) : Transition function

\( \chi \) : Initial state

\( E \) : Set of edges

\( \lambda \) : Output function
The usual definition of a finite automaton is as follows:

\[
FA(x) \text{ if } x = (Q, I, F, \Sigma, \Delta)
\]

\[
Q := \text{finite set of states}
\]
\[
I := \text{set of initial states}
\]
\[
F := \text{set of final states}
\]
\[
\Sigma^* := \text{input words}
\]
\[
\Delta := \text{set of transitions}
\]

If one replaces/ substitutes the vertices by states, the edges with action expressions by a transition with an input word then one gets an automaton. Finally, if one extends the structure of the automaton by the set of property-expressions \( \Pi \) as follows: \((Q, I, F, \Sigma, \Pi, \Delta)\) and with \( \lambda : Q \rightarrow 2^\Pi \), then one has an automaton with finite sets of properties attached to each state. We call a finite automaton with such a property extension a property-extended finite automaton (PFA).

**Simulation of Actor Story**: With this definition one has an extended automaton \( \alpha^+ \in \{x | PFA(x)\} \) as an automaton who being in state \( v \) recognizes an action expression \( \epsilon \in \chi \) and generates as follow-up state that state \( v' \), which is constructed out of state \( v \) by the encoded deletions and/or creations of properties given as property-expressions from \( \Pi \). All state-transitions of the automaton \( \alpha^+ \) from a start-state to a goal-state together are called a run \( \rho \) of the automaton. The set of all possible runs of the automaton is called the execution graph \( \gamma_{\text{exec}} \) of the automaton \( \alpha^+ \) or \( \gamma_{\text{exec}}(\alpha^+) \).

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 \( \gamma^+ \).

**Simulation of Actor Story with Actor Models**: Until now only that case has been described, where the actor story has been simulated without the actor models. A difference between the actor story and the actor models is given by the fact that the actor models are ‘from the beginning’ formal structures describing an algorithm. Thus it is only a question of convention to use a language \( L_{AM} \) which fits to an intended automaton. Additionally one needs some assisting algorithm to map the different actor model algorithms/ automata with the automaton of the actor story. Done in
the right way this ends up in a complete 'automatic conversion process' from actor story and actor models to a final simulation.
Another helpful process is the process named $\nu$. It translates a behavior model $M_{SR}$ with the aid of a temporal logic language $L_{TL}$ and an appropriate algorithm $\alpha$ into an algorithmic verification model $M_{\nu}$, which can compute the occurrence or non-occurrence of a certain property in the space of possible states of the behavior model. This capability of deciding the occurrence or non-occurrence of certain properties is especially helpful in the case of non-functional requirements.

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. This topic of 'Non-Functional Requirements (NFRs)' as well as 'Functional Requirements (FRs)' and their relationship is a hot topic in systems engineering and did not have a complete solution until now. The general problem is how to 'represent' the NFRs in a way, that these can be handled in the overall system. We have to demonstrate here one new approach to overcome the known problems.

This chapter describes how one can translate the logical specifications of the actor story and the actor models into a physical object with a physical appearance functioning as a possible physical interface assisting a possible physical executive actor as an intended user.
Usability Testing

As the preceding chapter about physical design 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 $\mathcal{UT}$ so far one realizes a mapping of given demonstrators $D$ into a set of usability values $V$ as follows $\mathcal{v}_{\mathcal{UT}} : D \rightarrow D \times V$. 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 $\mathcal{v}_{\mathcal{UT}}$ 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 $\{(D_1, V_1), (D_2, V_2), \ldots, (D_m, V_m)\}$ one has to define a measure $\mu : 2^{V^n} \rightarrow V^n$ for the assumed finite many $n$-dimensional values $\{V^n_1, V^n_2, \ldots, V^n_m\}$ to compare these values and identify for this set an optimal value. Thus $\mu(V^n_1, V^n_2, \ldots, V^n_m)$ computes a certain $V^n_i \in \{V^n_1, V^n_2, \ldots, V^n_m\}$.

Applying this measure to the set $\{(D_1, V_1), (D_2, V_2), \ldots, (D_m, V_m)\}$ 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
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 \(\{T_1, \ldots, T_n\}\) planned to mimicking the intended executive actor.

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

4. Compute the distance of each \(AAR_i\) 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 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 - 0 - +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.
10
AS-AM Philosophy

As explained in the introduction the 'Philosophy of the AAI-Expert' is centering around the findings of modern Biology and Psychology. Its aim is to explain why a human expert is able to use a formal language, here the set theoretical language $L_e$, to talk about his experiences of the empirical world. What Biology and Psychology are telling us is that the communication of the experts is grounded in their cognitive machinery embedded in their brains. Because the human brain in the body is not directly interacting with the outside world but mediated by sensors and actuators the brain 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. For more details see chapter 10 'AS and AM Philosophy'.

One conclusion from these considerations is that the reality for a human person is basically given as a stream of neural events, partially translated into phenomena of the consciousness, 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.

This, to understand how an AAI-expert perceives his world, generates internal models, and how he is communicating with others, this is the subject for a philosophical grounding of the preceding AAI analysis theory.

In case of the actor story we had introduced different modes to represent possible meanings with symbolic expressions which have as their primary point of reference the 'mental ontology' $DAT_{ontol}$ of the AAI experts. While the mental ontology is assumed to be 'the same' for all different modes of symbolic articulation\(^1\), 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' $L_{pict}$ 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 de-coding 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 $L_{pict}$ is sufficient. What is needed is a ‘formal language’ $L_{\epsilon}$ which 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!

REMARK: The following text has to be rewritten to fit to the topic of philosophical grounding.

The Logical Space

Looking back to all these new concepts and complex relationships the figure 10.1 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.

Creating the Symbolic Space

To speak about the structure of the symbolic space, its shape, its structure, is of limited use as long it is not clear, how one can generate such a symbolic space within a systems engineering process (SEP).

Here it is assumed that there exists a structured process of symbolic space creation. This presupposes that every participating AAI expert has a set of mental models (MMs) of his world view which represent for the AAI expert the important properties of the known world.(See for this ² and

\[ \text{Figure 10.2: Creation of the symbolic space by AAI-experts} \]

It is further assumed that the AAI experts have some languages in common which allows the production of a symbolic space, which can be shared by the whole team. Because this symbolic space is external to all participants it can reflect back to the different authors and thereby synchronizing the different individual mental models. In a finite set of modifications occurring as an iterative process evolves the symbolic space first as an actor story (AS), then as a finite set of actor models (AMs).

Creating such a symbolic space 'by hand', 'manually' is possible and should always be possible in principle, but for more advanced symbolic spaces one needs a support by specialized SW-tools or even – in the long run – by specialized AI programs.

**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 10.3 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.

Multiple Actor Stories

Until now we have only looked to a single actor story (AS), that, which just has to be created as a new one. But the intended actors as real actors in a real world usually are living in more than one actor story either at the same time or in different time slices, with a different actor story in a different time slice. Nevertheless because a real person has a physical body
which needs different physical resources, physical transportations, physical communications and more there are several interactions between different actor stories. Thus, to make an actor story 'stable' one has to include these different kinds of interactions with accompanying actor stories from other stakeholders.
11

Looking Forward

Having completed the AAI analysis according to the schema 1.1 one has to continue the overall systems engineering process with logical design, implementation, final validation and then deployment, to mention the minimal layout of such a process.\footnote{For a more detailed description of these processes after the AAI analysis see the paper Erasmus and Doeben-Henisch (2011) Louwrence Erasmus and Gerd Doeben-Henisch. A theory of the system engineering management processes. In ISEM 2011 International Conference. ISEM, 2011b. Conference 2011, September 21-23, Stellenbosch, South Africa}

With this new version of a complete theory for AAI analysis (AAIA-TH) a new option becomes available: either you can stop the systems engineering process with the simulators at hand which allow lots of testing, of simulations, of learning, even of fun when used for gaming, or you can continue the system engineering process if you want to built some machinery which departs from pure simulation into the realm of physical devices (‘real car’, ‘real airplane’, ‘real power plant’, ‘real city’, ...).

But there is a third option too: with the new ‘wave’ ‘internet of things (IoT)’ a paradigm is pushed where parts of the physical world are connected with sensors and actuators such that some algorithm can control these real-world parts by sensing or acting. In case of the simulators of the new AAI analysis theory one can combine this IoT-paradigm with the AAIA theory. This would expand the idea of Industry X.0 to the idea of World X.0; this would represent the full paradigm of a ‘digitization of the world’. This option seems to be highly promising. A good motivation why the usual paradigm of a ‘Smart City’ is not sufficient and needs an improvement can be get from the research from Mila Gascó-Hernandez (2016)\footnote{Mila Gascó. What makes a city smart? lessons from barcelona. In 2016 49th Hawaii International Conference on System Sciences (HICSS), pages 2983–2989, Jan 2016. DOI: 10.1109/HICSS.2016.373}, (2018)\footnote{Mila Gascó-Hernandez. Building a smart city: Lessons from barcelona. Commun. ACM, 61(4):50–57, March 2018. ISSN 0001-0782. DOI: 10.1145/3117800. URL http://doi.acm.org/10.1145/3117800}.

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/
Appendix: Actor Story Example

An Example Problem

As described in the chapter 3 'Actor Story (AS)' the starting point for an actor story is a problem document $D_P$ which describes in a first way which kind of a problem a stakeholder wants to be solved.

In this example we assume a small German city with a mayor whose primary goal it is, to organize a process which enables him to do a constructive planning of the future with all necessary groups of the city which have to be convinced to cooperate. We call the members of all these different groups citizens although this can be elected community representatives, members of technical departments, and more.

Inspired by modern research this mayor has learned that the enabling of a cooperation must be based on a common world model which is accepted between all participating citizens.

Thus the mayor wants to have a procedure which can assists him and the citizens to elaborate such common models and to assist to use these models in a productive way.

Outline of a possible Actor Story, Version 1

The elaboration of an actor story with a team of AAI experts enriched with other experts can in itself be understood as a communication process, whose outcome is a first version of an actor story (AS).
The whole idea of the following actor story can be summarized as follows (cf. figure 11.1):

1. A stakeholder – in the actual case a mayor – wants to cooperate with other citizens based on commonly shared models of the city. Starting point are questions $Q$ he and the others want to be answered.

2. All the questions together are bundled as a problem $P$ which shall be solved.

3. Given a problem document $D_P$ the stakeholder wants to be able to construct together with others a model shaped as a set of rules connecting different factors in a procedural (functional, algorithmic) way. The model should be interpretable as a possible answer to the questions constituting the problem.

4. It shall be possible to run a simulation with the model. The simulation shall show all possible states the model can describe in a future following some start state which is assumed to represent the actual state of the town with regard to some selected factors.

5. To make the experience of the model more intensive it should be able to run the simulation in gaming mode. This requires that the simulation is interactive and that more than one player can interact in the same time. By the actions a player can influence the simulation during run-time. Thus the process will show possible states of the model influenced by the behavior of the different players.

6. With the aid of a simulation as well with the aid of the gaming mode the user of these two modes can foresee lots of states which can follow in this model in some future.

7. Having such forecasts the user can try to verify these forecasts by exploring the real city with regard to these predicted upcoming states. In some cases the verification can be simple by direct inspection; in other cases data may not be available, only in the future itself.

8. As long as the predictions agree with the observable data of the real city everything looks fine and the model can be assumed to be ‘adequate’, ‘appropriate’, to be ‘verified, to be ‘true’. In other cases it can be either ‘open’ whether the model will be appropriate or the observable data are not in agreement with the model. In these cases either the observable data can be wrong (has to be checked) or the model does not fit because it is in-adequate, it is ‘false’.

9. If a model seems to be false one can either modify the given model to ‘improve’ it for a better fitting with the real city or one can adhere to the model and start to change the real city according to the model. In the last case the model is used as an inventive tool to detect new and more preferable states for the city.

Textual Actor Story (TAS), version 1

In the following text the outline for a possible actor story in textual mode will be refined step by step.
**QUESTIONS:** Idea: a stakeholder – in the actual case a mayor – wants to cooperate with other citizens based on commonly shared models of the city. The starting point shall be questions (Q) the mayor and the others want to be answered. For this a real workshop or a virtual meeting between a mayor and some citizens is organized: There are some known factors describing properties of the city, e.g.: the number of inhabitants \( N \) classified according to gender (M,F,X), age, the average birth-rate \( BR \) from the last 10 years, the average death-rate \( DR \), the positive migration rate \( MIG+ \), and the negative migration rate \( MIG− \). Given these data the citizens and the mayor raised some questions, e.g.:

(a) What will be the number of inhabitants in the future in \( n \)-many years \( N_n \), distinguished between age and gender? What are the actual numbers \( N_{\text{now}} \)?

(b) How many square-meters living space \( SPACE_{\text{living, sqm}, n} \) will be needed for this population? How many are actual available \( SPACE_{\text{living, sqm, now}} \)?

(c) How much energy measured in kilowatt hour \( E_{n, \text{Wh}} \) will be needed for this population? How much is actual available \( E_{\text{now, Wh}} \)?

(d) How many liter drinkable water \( WATER_{\text{drink, l}, n} \) will be needed for this population? How many are actual available \( WATER_{\text{drink, l, now}} \)?

(e) How many liter garbage \( GARBGl_{n} \) will be produced by this population? How many are actual produced \( GARBGl_{\text{now}} \)?

(f) How many persons have to be transported \( TRANSF_{\text{n}} \) during the day? How many are actually transported \( TRANSF_{\text{now}} \)?

(g) How large will be the revenues of the city \( REV_{\text{city, n}} \) in the next years? How large is it actual \( REV_{\text{city, now}} \)?

(h) How large will be the expenditures of the city \( EXPDT_{\text{city, n}} \) in the next years? How large is it actual \( EXPDT_{\text{city, now}} \)?

(i) How are the revenues related to the before mentioned factors?

(j) How are the expenditures related to the before mentioned factors?

All the known factors as well as the raised questions are written into a question document \( D_Q \).

**PROBLEM:** Idea: All the questions together with the known factors given in the questions document \( D_Q \) are then bundled as a problem (P) which shall be solved. Realization: Additionally to the known factors and the raised questions the problem statement poses some additional constraints, how these facts shall be available for the intended users as the executive actors. This can be e.g. as follows:

(a) The known factors can be collected in a set of properties called the actual state of the city \( CITY\text{-GIVEN} = \{ N_{\text{now}}, M_{\text{now}}, F_{\text{now}}, X_{\text{now}}, BR, DR, MIG+, MIG− \} \)

(b) The unknown factors can be collected in a set of properties called the unknown state of the city

\( CITY\text{-UNKNOWN1} = \{ N_n, M_n, F_n, X_n, BR, DR, MIG+, MIG− \}, \)

\( CITY\text{-UNKNOWN2} = \{ SPACE_{\text{living, sqm, n}}, SPACE_{\text{living, sqm, now}}, E_{\text{now, Wh}}, E_{n, \text{Wh}} \}, \)

\( CITY\text{-UNKNOWN3} = \{ WATER_{\text{drink, l, n}}, WATER_{\text{drink, l, now}}, GARBGl_{n, \text{now}}, GARBGl_{n} \}, \)
CITY-UNKNOWN3 = \{\text{TRANSF}_\text{now}, \text{TRANSF}_n, \text{REV}_{\text{city,now}}, \text{REV}_{\text{city,n}}, \text{EXPDT}_{\text{city,now}}, \text{EXPDT}_{\text{city,n}}\},
with \(n > \text{now}\).

(c) The mayor and the citizens shall be able to enter these factors during modeling on the screen of their mobiles together with the questioned relations. They have the additional option to select between simulation mode or gaming mode. Depending on this selection the system will start the selected mode. It should be possible to leave the simulation or gaming mode and to return to the modeling or even to the problem or question mode.

MODEL: Given a problem document \(D_P\) the stakeholder wants to be able to construct together with others a model shaped as a set of rules connecting different factors in a procedural (functional, algorithmic) way. The model should be interpretable as a possible answer to the questions constituting the problem.

SIMULATION: It shall be possible to run a simulation with the model. The simulation shall show all possible states the model can describe in a future following some start state which is assumed to represent the actual state of the town with regard to some selected factors.

GAMING: To make the experience of the model more intensive it should be able to run the simulation in gaming mode. This requires that the simulation is interactive and that more than one player can interact in the same time. By the actions a player can influence the simulation during run-time. Thus the process will show possible states of the model influenced by the behavior of the different players.

FORECASTS: With the aid of a simulation as well with the aid of the gaming mode the user of these two modes can foresee lots of states which can follow in this model in some future.

DATA VERIFICATION: Having such forecasts the user can try to verify these forecasts by exploring the real city with regard to these predicted upcoming states. In some cases the verification can be simple by direct inspection; in other cases data may not be available, only in the future itself.

EVALUATION: As long as the predictions agree with the observable data of the real city everything looks fine and the model can be assumed to be ‘adequate’, ‘appropriate’, to be ‘verified’, to be ‘true’. In other cases it can be either ‘open’ whether the model will be appropriate or the observable data are not in agreement with the model. In these cases either the observable data can be wrong (has to be checked) or the model does not fit because it is in-adequate, it is ‘false’.

MODIFICATION: If a model seems to be false one can either modify the given model to ‘improve’ it for a better fitting with the real city or one can adhere to the model and start to change the real city according to the model. In the last case the model is used as an inventive tool to detect new and more preferable states for the city.
Bibliography


Index

1st-person view, 20
3rd-person view, 20

AAI, 14
AAI analysis, 17
AAI analysis formalized, 15
AAI analysis theory (AAIA-TH), 63
AAI check, 17
AAI-analysis structure, 14
AAI-expert philosophy, 16
AAR, 29, 61
action as mapping, 28
action definition, 28
actor, 19, 23, 34, 57
actor as iosys, 34
actor induced actor requirements, 29
actor model, 15, 19, 33
actor story, 15, 19
actor story (AS), 19
actor-actor interaction, 13
actor-environment interaction, 34
adaptiv meaning, 24
algebraic structure, 34
algorithmic verification, 49
AM, 33
AS and AM, 30
AS as graph, 40
AS example no.1, 65
AS-AM Philosophy, 57
assistive actor, 23
assitive actor, 18
automatic verification, 21
automaton, 21
automaton execution graph, 46

basic typology IO-systems, 36
behavior error rate, 54
behavior model, 15, 18, 19
best demonstrator, 53
Blntf, 34
Biology, 57
brain, 20, 24, 57
building assistive actors, 60
Card, 38
change, 19, 23
change event, 29
change language, 21
citizens, 65
city, 65
classify AAR, 54
common world model, 65
communication, 20
confirmation, 24
conscious structure, 26
consciousness, 20
continuous learning, 37
creation symbolic space, 59
curve of satisfaction, 54
decoding from symbols, 20
design interface, 16
deterministic behavior, 39
deterministic output set, 37
digitization of the world, 63
domain of reference, 20
dynamics of learning behavior, 54
empirical behavior function, 33
empirical matters, 24
empirically motivated, 38
encoding into symbols, 20
enhanced graph, 29
environment, 18, 34
evaluating verifications, 66, 68
everyday language, 21, 23, 57
example electronic door, 39
executing actor, 23
executive actor, 18
expected behavior, 43
extended graph, 27
extended ordered graph (EOG), 45
feasible task, 23
finding new states, 66, 68
finite automaton (FA), 46
fixed output set, 37
foreseeing the future, 66, 68
forgetting curve, 54
forgetting time, 43
formal language, 57, 58
functional requirements, 20
future, 19
fuzzy predictions, 37
gaming mode, 66, 68
generate AAR, 54
generate output, 33
getting rewards, 37
GOAL, 39
GOMS model, 39
GOMS rule format, 39
graph, 26
graph language, 21
graph-automaton conversion, 47
heuristic guide, 33
human actor, 61
human-computer interaction, 13
human-machine interaction, 13
INCOSE, 13
incremental output set, 37
inner model of outside world, 57
inner states, 44
innovation, 66, 68
input-output system, 19, 34
intelligence, 43
interface, 18
interface design, 29, 36
interface requirements, 29
internet of things (IoT), 63
introduction, 13
IOSYS, 34
language fuzziness, 21
language modes, 21
learning behavior, 43
learning curve, 54
learning IOSYS, 37
learning time, 43
license, 4
logical space, 58
looking forward, 63