. . . THE ABILITY OF 'MATTER' TO ENABLE A BRAIN WITH A CONSCIOUSNESS, WHICH CAN CONSTRUCT A 'THEORY OF A WHOLE UNIVERSE' IS AN OUTSTANDING PHENOMENON. TO BE A COMPLETE THEORY THE THEORY-GENERATING BRAIN AND ITS THEORY ITSELF SHOULD BE PART OF THE 'THEORY OF A WHOLE UNIVERSE'. BUT THIS IS BY PRINCIPAL REASONS NOT POSSIBLE (GOEDEL 1931, HAWKING 2002 [1]). BUT THE ACTOR-ACTOR INTERACTION PARADIGM IN THIS BOOK DOES INCLUDE THE THEORY PRODUCER IN THE THEORY, ... IS THIS CRAZY?

GERD DOEBEN-HENISCH

[1] Kurt Goedel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, i. *Monatshefte fuer Mathematik und Physik*, 38:173–98, 1931; and Stephen Hawking. Gödel and the end of physics, 2002. URL http://www.hawking.org.uk/godel-and-the-end-of-physics.html

GERD DOEBEN-HENISCH, LOUWRENCE ERASMUS, ZEYNEP TUNCER

ACTOR ACTOR INTERACTION [AAI] WITHIN SYSTEMS ENGINEERING (SE)

AN ACTOR CENTERED APPROACH TO PROBLEM SOLVING IN ENGINEERING
COMBINING ENGINEERING AND PHILOSOPHY

VERSION 2.OCTOBER 2018

UFFMM.ORG

PUBLISHED BY UFFMM.ORG

UFFMM.ORG, ISSN 2567-6458

First printing, October 2018

Contents

List of Figures

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 has three parts. Part I gives you an introduction into the theoretical view of the new actor-actor analysis within systems engineering. In part II we apply this theory to a process model which describes how one can realize the theory in the real world. In part III we give a first outline of a possible software demonstrator. This last part is purely experimental and should be replaced in the future by a more professional implementation.

Every part of 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 theory and application stuff for the community. It is also planned to provide all the software on the website `https://emerging-mind.org/`

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[2], many intensive discussions between Doeben-Henisch and Erasmus about Systems engineering since 1999, a paper written by Doeben-Henisch and Wagner 2007 [3] with ongoing discussions since then, a lecture by Doeben-Henisch about formal specification and verification in 2010 [4], two papers by Erasmus and Doeben Henisch in 2011 [5], 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 time span 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

[2] He died 10.May 2018.

[3] G. Doeben-Henisch and M. Wagner. Validation within safety critical systems engineering from a computational semiotics point of view. *Proceedings of the IEEE Africon2007 Conference*, pages Pages: 1 – 7, 2007. DOI: 10.1109/AFRICON.2007.4401588

[4] Gerd Doeben-Henisch. *Formal Specification and Verification: Short Introduction*. Gerd Doeben-Henisch, 2010

[5] Louwrence Erasmus and Gerd Doeben-Henisch. A theory of the system engineering process. In *9th IEEE AFRICON Conference*. IEEE, 2011a; and 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

(unpublished). Since SS2016 ongoing discussions and a cooperation with Zeynep Tuncer about the topics of the book.

Part I

Theory

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 an 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)'.

[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.hcibib.org/), which has a rich historical section too (see: http://www.hcibib.org/hci-sites/history).
Richard W. Pew. Introduction. Evolution of human-computer interaction: From memex to bluetooth and beyond. In J.A. Jacko and A. Sears, editors, *The Human-Computer Interaction Handbook. Fundamentals, Evolving Technologies, and emerging Applications*. 1 edition, 2003; and Jonathan Grudin. A Moving Target: The Evolution of HCI. In A. Sears and J.A. Jacko, editors, *The Human-Computer Interaction Handbook. Fundamentals, Evolving Technologies, and emerging Applications*. 2 edition, 2008

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.

[2] For a first introduction cf. INCOSE (2015)
INCOSE. *SYSTEMS ENGINEERING HANDBOOK. A GUIDE FOR SYSTEM LIFE CYCLE PROCESSES AND ACTIVITIES*. John Wiley & Sons, Inc, Hoboken, New Jersey, 4 edition, 2015

In systems engineering (cf. figure 1.1) it is common to assume an *expert* as part of a *systems 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 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.

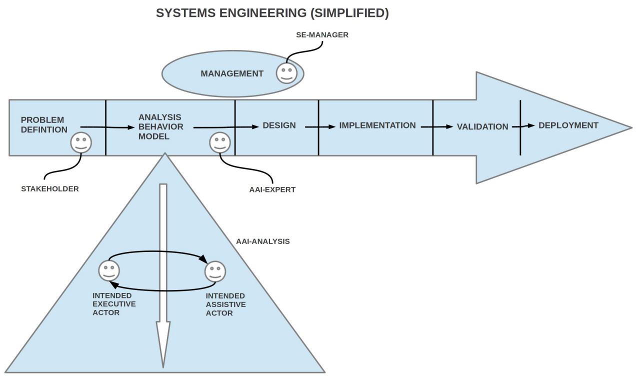

Figure 1.1: Engineering process with diferent kinds of actors

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].

[3] For the historical motivation of this approach see the before mentioned papers from Erasmus, Doeben-Henisch, and Wagner.

$$
\begin{aligned}
AAIA(x) \quad & iff \qquad (1.1)\\
x = \langle A, D, D_P, D_{AAR}, M_{SR}, M_\sigma, M_\nu, M_{DIntf}, M^*_{DIntf}, \delta \rangle & \\
A &:= Set\ of\ actors\\
D &:= Set\ of\ documents\\
D_P &:= Set\ of\ problem\ documents\\
D_{AAR} &:= Set\ of\ actor\ requirements\ documents\\
M_{SR} &:= Set\ of\ behavior\ models\\
M_\sigma &:= Set\ of\ simulator\ models\\
M_\nu &:= Set\ of\ algorithmic\ verification\ models\\
M_{DIntf} &:= Set\ of\ real\ interfaces\\
M^*_{DIntf} &:= Set\ of\ optimized\ real\ interfaces\\
\delta &= \alpha \otimes \beta \otimes \sigma \otimes \pi \otimes \gamma \otimes o\\
\alpha &: A \times D \longmapsto D_P
\end{aligned}
$$

$$
\begin{aligned}
\beta &: A \times D \times D_P \longmapsto M_{SR} \\
\sigma &: A \times D \times M_{SR} \longmapsto M_\sigma \\
\nu &: A \times D \times M_{SR} \longmapsto M_\nu \\
\gamma &: A \times D \times M_{SR} \times D_{AAR} \longmapsto M_{DIntf} \\
\omega &: A \times D \times M_{SR} \times D_{AAR} \times M_{DIntf} \longmapsto M^*_{DIntf}
\end{aligned}
$$

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 α 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 β 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_ϵ 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 acor story AS_ϵ in mathematical mode.

Based on the mathematical mode of an actor story AS_ϵ one can convert the actor story AS_ϵ with an algorithm into an *automaton* M_α which can be run on an appropriate computer as a *simulation*. The combination of this automaton M_α with an appropriate computer we call a *simulator model* M_σ. The whole process preparing a behavior model M_{SR} as a simulator model is called σ.

Another helpful process is the process named ν. It translates a behavior model M_{SR} with the aid of a *temporal logic language* L_{TL} and an appropriate algorithm α into a *algorithmic verification model* M_ν, 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 an 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 γ one has to use knowledge from the *actor-actor induced requirements (AAR)* as well as knowledge from Psychology to design a physical appearance of the assistive actors M_{DIntf} 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 ω-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_{ϵ}, 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.

Thus, 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.

2.1 Problem Document

According to the schema 1.1 the first sub-process is given by the process '$\alpha : A \times D \longmapsto D_P$'. 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.

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

G. Doeben-Henisch and M. Wagner. Validation within safety critical systems engineering from a computational semiotics point of view. *Proceedings of the IEEE Africon2007 Conference*, pages Pages: 1 – 7, 2007. DOI: 10.1109/AFRICON.2007.4401588

2.2 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.

2.3 Behavior Model

Following the schema 1.1 further we meet the next sub-process called beta $\beta : A \times D \times D_P \longmapsto M_{SR}$. 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_ϵ 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 acor story AS_ϵ 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)

3.1 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*, and at least one *assistive and executive actor*.

FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS : The actor story represents all necessary *functional requirements (FR)*. For *non-functional requirement (NFR)* see chapter 7 'Algorithmic Verification'.

3RD PERSON VIEW: The *point of view* underlying the description of an actor story AS is the so-called 3^{rd}-person view. This means that all participating objects and actors are described from their *outside.* If an actor *acts* and changes some property through it's action it is not possible in a 3^{rd}-person view to describe the inner states and inner processes, that enabled the actor to act and why he acts in this way. An inner state corresponds to what often is called the 1^{st}-person view. To include this additional perspective and thereby to overcome the limits of a 3^{rd}-person view one has to construct additional models called *Actor Models (AMs)* as described in the the chapter 5.

BEHAVIOR MODEL: 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 chapter 5.

3.2 Actor Story, Domain of Reference

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 being not 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.

All these mentioned terms like 'task', 'state', event', 'actor' and the like constitute what usually is called the *domain of reference* D^{Ref}. This domain of reference is located in the *internal states* of an actor, which in turn are located in the *consciousness* of a person as part of the *brain*. The consciousness is closely related with *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.

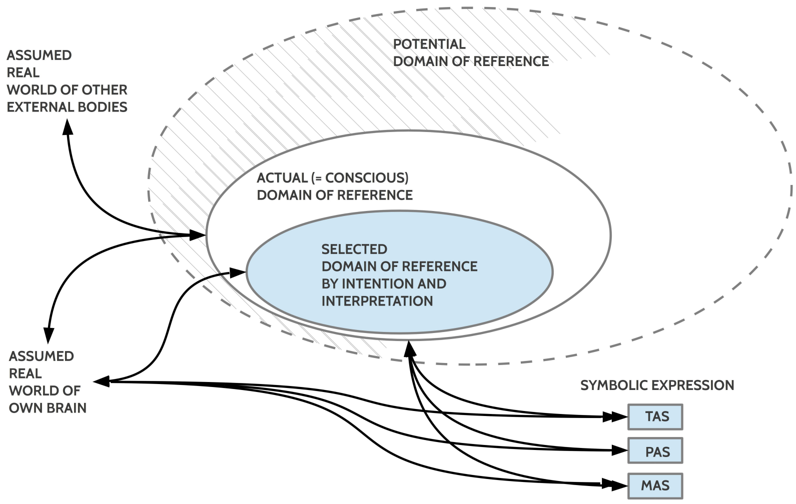

Figure 3.1: Domain of reference as a field of interconnected states

Thus the so-called 'domain of reference' D^{Ref} is a key concept in the construction of an actor model. As figure 3.1 shows the *actual* consciousness D^{Ref} presents only a subset of a nearly infinite field of *potential* references $D^{Ref,\omega}$. And every symbolic expression $e \in L_\omega$ with a *learned interpretation* τ selects by this interpretation a learned subset of the consciousness as the actual *intended meaning* of the used expression e, written as

$$\tau^{-1} \quad : \quad L_\omega \longmapsto D^{Ref} \tag{3.1}$$

In the reverse order, if one has some 'intention' ι which shall be communicated, then the mapping has the format:

$$\tau \quad : \quad D^{Ref} \times \iota \longmapsto L_\omega$$

This domain of reference is embedded in the working of the brain which (i) maps some properties of an assumed outside world of other bodies into internal brain states which can become 'conscious' as well (ii) maps some properties of the brain itself (e.g. memory contents) into the 'conscious states'. Moreover (iii) manages the brain different kinds of expression systems (like TAS, PAS, and MAS)[1] as well as (iv) interpretation relations between the expression systems and the potential domain of reference. In special cases can the expression systems themselves being part of the domain of reference but in the 'normal' case they are part of an interpretation relation τ' where some *intention* ι has selected some subset of the actual domain of reference to be mapped by the interpretation function τ.

[1] See next chapter.

3.3 Cognitive Structure of an AAI-Expert

The conceptual framework so far induces minimal assumptions about the properties of an AAI expert doing all this work. Thus we have (in a simplified structure):

$$
\begin{aligned}
EXP(x) \quad & iff \quad x = \langle B, C, M, D^{Ref}, \tau, \tau^{-1}, L_{\omega}, \Omega \rangle \\
B \quad & := \quad Brain \\
C \quad & := \quad Consciousness; C \subseteq B \\
M \quad & := \quad Memory; M \subseteq B \\
L_{\omega} \quad & := \quad Some\ language \\
\Omega \quad & := \quad \Omega \subseteq \{x | x \in L_{\omega}\} \\
D^{Ref} \quad & := \quad Domain\ of\ reference \\
D^{Ref} \quad & \subseteq \quad B \cup C \cup M \\
\tau \quad & : \quad D^{Ref} \longmapsto \Omega \\
\tau^{-1} \quad & : \quad \Omega \longmapsto D^{Ref}
\end{aligned}
\tag{3.2}
$$

The τ-function represents different kinds of mappings depending from a set of expressions from languages L_{ω}. Below you can see examples of different languages like $L_0, L_{prop}, L_x, L_{pict}, ...$

The main message here is that if two different AAI experts a and b want to communicate and they share the expressions of a certain language L_{omega} then they have no direct access to the meaning structure of the other expert. Thus they have to develop and exercise certain conventions how to secure that they are using the same meaning when they are using the same expressions.

3.4 Coordination by Communication

Assuming the internal (cognitive) structure of an AAI expert as described before in the 'EXPERT' definition 3.2 one can infer that a brain a, which wants to *coordinate* its contents with some other brain b has to enable some kind of *communication* which allows a continuous *encoding* of its internal

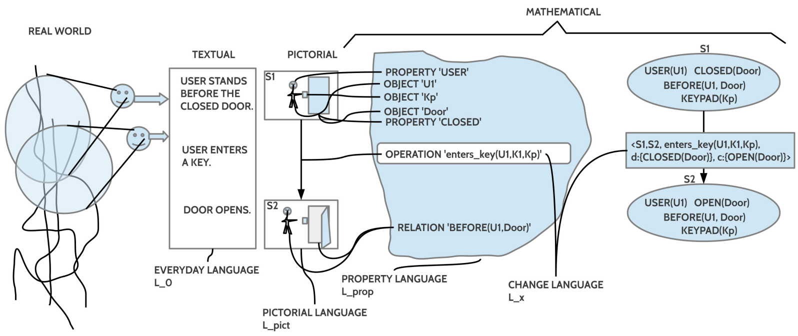

Figure 3.2: Translating the individual world views into three different languages

states into the *expressions* of some *language* L_ω and simultaneously a *decoding* of these expressions into the internal states of brain b.

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: $sndSymb : L_{0,snd} \longmapsto L_{0,symb}$ and $symbSnd : L_{0,symb} \longmapsto L_{0,snd}$. 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 a re-mapping from the written symbols into the utterance situation is 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 3.2 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_{mth}. 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 in restricted sense 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_{mth} 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* $L_{mth,graph}$, a language of *properties* $L_{mth,prop}$, and a language of *changes* $L_{mth,x}$.[2]

The '*mathematical language*' L_{mth} can easily be used for logical proofs,

[2] Later in this book it will be shown that the mathematical language L_{mth} can be enhanced by even more languages.

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.[3] Additionally one can apply automatic verification for selected properties, e.g. for non-functional requirements.[4]

[3] For simulation see chapter 6.

[4] For automatic verification see chapter 7.

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.[5]

[5] 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.

4.1 Textual Actor Story (TAS)

For an actor story AS in the *textual mode* – here called 'textual actor story (TAS) – the AAI experts are assumed to have – according to the definition of an EXPERT in 3.2 – a cognitive structure where the general language L_{ω} is substituted by an everyday language L_0. The default language here is English L_{EN}:

$$
\begin{array}{rcl}
EXP_{txt}(x) & iff & x = \langle B, C, M, D^{Ref}, \tau_{txt}, \tau_{txt}^{-1}, L_0, TAS \rangle \\
B & := & Brain \\
C & := & Consciousness; C \subseteq B \\
M & := & Memory; M \subseteq B \\
L_0 & := & Everyday\ language \\
TAS & := & A\ text\ written\ in\ everyday\ language \\
D^{Ref} & := & Domain\ of\ reference \\
D^{Ref} & \subseteq & B \cup C \cup M \\
\tau_{txt} & : & D^{Ref} \times L_0 \longmapsto TAS \\
\tau_{txt}^{-1} & : & TAS \longmapsto D^{Ref}
\end{array}
\quad (4.1)
$$

Thus the 'textual' mapping function τ_{txt} translates internal conscious states constituting the domain of reference with the aid of a set of expressions L_0 into some (symbolic) TAS-expressions and the inverse textual meaning function τ_{txt}^{-1} maps in reverse order symbolic TAS-expressions back to the domain of reference.

4.1.1 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 ++++

4.2 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 in a pictorial actor story, which has sufficient resemblances with the intended meaning, it is possible to *show* parts of the encoded meaning more directly. Therefore a pictorial actor story can *substitute* the intended meaning to some degree if the pictorial language provides pictures which are structurally sufficient similar to the perceived visual structure of the observer.

$$
\begin{aligned}
EXP_{pict}(x) \quad & iff \quad x = \langle B, C, M, D^{Ref}, \tau_{pict}, \tau_{pict}^{-1}, L_{pict}, PAS \rangle \quad (4.2)\\
B \quad & := \quad Brain\\
C \quad & := \quad Consciousness; C \subseteq B\\
M \quad & := \quad Memory; M \subseteq B\\
L_{pict} \quad & := \quad Pictorial\ language\\
PAS \quad & := \quad A\ story\ written\ in\ pictorial\ language\\
D^{Ref} \quad & := \quad Domain\ of\ reference\\
D^{Ref} \quad & \subseteq \quad B \cup C \cup M\\
\tau_{pict} \quad & : \quad D^{Ref} \times L_{pict} \longmapsto PAS\\
\tau_{pict}^{-1} \quad & : \quad PAS \longmapsto D^{Ref}
\end{aligned}
$$

Because the intended meaning as part of the domain of reference, 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.[1]

[1] 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.

4.2.1 PAS Example: Open Door

The figure 4.1 shows an example of possible simple pictorial actor story. This PAS uses the same 'meaning' as in the before mentioned case of a textual actor story. As one can (possibly) see there exist a strong resemblance between the PAS and the 'imaginations' which are caused by the reading of the textual version. This pushes the motivation to construct generally a pictorial-textual mapping called a *picture-text lexicon (PTLex)*. In the follow-up chapter about simulation such a picture-text lexicon could enhance

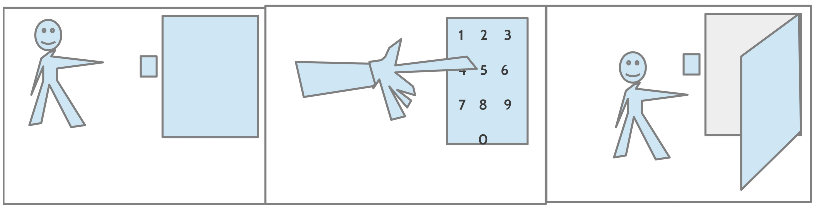

Figure 4.1: Simple pictorial actor story associated with the 'open door TAS'

the formal symbol-based simulations in a way which makes it far better 'understandable' for a human user than without such a picture dimension.

Figure 4.2 shows a simple example of a picture-text lexicon, here in the reverse order text-to-picture.

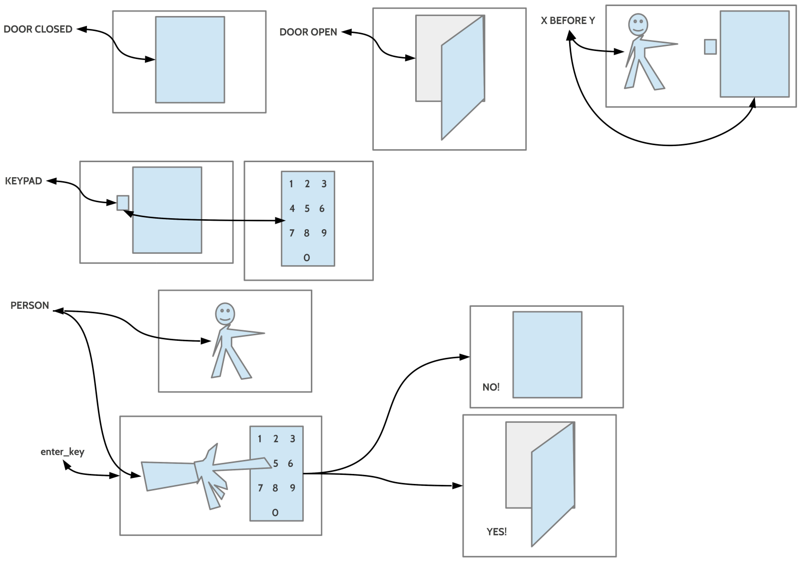

Figure 4.2: Informal example of a word-to-picture lexicon as starting point for a mapping from TAS into PAS

In this figure 4.2 the mapping from TAS to PAS as follows (internal to the expert) the following rules:

$$\begin{aligned} state &\leftrightarrow single\ picture\ frame && (4.3)\\ object \cup properties &\leftrightarrow shape \cup text && (4.4)\\ relation &\leftrightarrow shapes\ in\ neighbourship \cup text && (4.5)\\ action &\leftrightarrow state\ and\ followup\ states && (4.6)\end{aligned}$$

4.3 Mathematical Actor Story (MAS)

In the case of the mathematical actor story (MAS) we have a slightly more complex mapping. Generally it would be possible to rewrite the structure used before as follows:

$$\begin{aligned} EXP_{mth}(x) \quad &iff \quad x = \langle B, C, M, D^{Ref}, \tau_{mth}, \tau^{-1}_{mth}, L_{mth}, MAS\rangle && (4.7)\\ B \quad &:= \quad Brain \end{aligned}$$

$$
\begin{aligned}
C &:= Consciousness; C \subseteq B \\
M &:= Memory; M \subseteq B \\
L_{mth} &:= Mathematical\ language \\
MAS &:= A\ story\ written\ with\ a\ mathematical\ language \\
D^{Ref} &:= Domain\ of\ reference \\
D^{Ref} &\subseteq B \cup C \cup M \\
\tau_{mth} &: D^{Ref} \times L_{mth} \longmapsto MAS \\
\tau_{mth}^{-1} &: MAS \longmapsto D^{Ref}
\end{aligned}
$$

But the mathematical actor story (MAS) as such has a complex structure which is given in the following definition of an extended graph:

$$
\begin{aligned}
\gamma^{+}(g) &\ iff\ g = \langle V, E, \Pi, \Xi, \lambda, \epsilon \rangle \\
V &:= vertices \\
E &:= edges \\
E &\subseteq V \times V \\
\Pi &= 2^{L_{prop}}; Property\ expressions \\
\Xi &:= change\ expressions \\
\lambda &: V \longrightarrow 2^{\Pi} \\
\epsilon &: E \longrightarrow \Xi
\end{aligned} \tag{4.8}
$$

A mathematical actor story (MAS) is then to be understood as an element of the set of extended graphs:

$$
MAS \subseteq \{x | \gamma^{+}(x)\} \tag{4.9}
$$

4.3.1 Property Language

The set of properties Π is a subset of expressions defined by the *property language* L_{prop}. One can describe the property language L_{prop} as follows:

1. Expressions used as *names* for possible objects in the domain of reference D^{Ref} are written as strings of symbols from the ASCII-Alphabet starting with a capital letter like 'A', 'B', ... followed by small letters like 'a', 'b' ... mixed with 'digits' like '0', '1', '2', ... or a 'dash' like '-'. If there exists a corresponding object in D^{Ref} then it holds that the name is *sound* or *has a meaning*.
2. Expressions used as n-ary *operations* with n-many names as arguments refer to the domain of reference D^{Ref} and are written as strings of symbols from the ASCII-Alphabet starting with a lower letter like 'a', 'b' ... followed by more lower case letters or by capital letters like 'A', 'B', ... mixed with 'digits' like '0', '1', '2', ... or a 'dash' like '-'. If there exists corresponding objects in D^{Ref} then it holds that the operation is *sound*

or *has a meaning*. Usually operations require two states (q,q') where q' is a successor state of state q and there is at least one property in q' different from the set of properties in q.

3. A name is also called an *atomic term (AT)*. An operator with names as arguments is called a *complex term (CT)*. Atomic as well as complex terms are called *terms (TRM)*.

4. Expressions used as *predicates (PRED)* are written as strings of symbols from the ASCII-Alphabet containing capital letters like 'A', 'B', ... , but can include 'dashes' like '-'. Predicates with names (atomic terms) as arguments are here called *Properties (PROP)* or *property statements* which can be *valid* in the domain of reference, also called being *sound* or being *true*. If they are not valid in the domain of reference then they are called *not sound* or *not true*, which is used equivalently to being *false*. Predicates which have only one name as argument are also called *features (FEAT)* or *attributes (ATTR)*, those with more than one name as argument are called *relations (REL)*.

5. Predicate expressions which are preceded by a *negator symbol* '$\neg$' are called *negated statements (NST)*. If the predicate statement following the negator sign is called 'true' in D^{Ref} then the negated statement is called 'false' in D^{Ref}, and 'true' otherwise.

PREDICATES AND COMPLEX TERMS: Different to usual conventions in the usage of predicate-logic related languages[2] predicates and complex terms will here be used as *properties* although complex terms can not become true while predicates can. Statements with predicates can be decided directly as 'true' or 'not true', while the complex terms are in the domain of reference represented by pairs of succeeding states (q,q'). From this follows that a complex term as an operation is associated with a *distributed process* where a state q does *indicate* the occurrence of a complex term wich serves as a 'forcast' for a *change*, which becomes 'public' in a follow-up state which is showing the 'effect' of the operation. Such a pair of states is represented in a graph by an edge which is labeled with a *change statement* 'citing' the triggering operation.

[2] c.f. Nilsson (1998), Russell/ Norvig (2010)

Nils J. Nilsson, editor. *Artificial Intelligence*. Morgan Kaufmann, Menlo Park, 1998; and Stuart Russel and Peter Norvig. *Artificial Intelligence. A Modern Approach*. Universe Books, 3 edition, 2010

Knowing this one can define the set of property-expressions Π as follows:

$$\Pi \quad \subseteq \quad \{x|PROP(x) \vee CT(x)\} \tag{4.10}$$

Example: When in a state q an actor U1 keys in a code K into a keypad Kp – expressed as 'enterKey(U1,K,Kp)' – then this causes a follow-up state q' with a defined effect. The connecting edge has attached a change-statement (see below).

4.3.2 Change Language

In the context of an actor story there are basically different states in some successive order. A *follow-up state* q' to a preceding state q has at least one property *different* to the preceding state. Whether this *different property* is either a *new* property or a *disappearing* property depends either on an *action* which *succeeds* or depends on a some *event* which *occurs*.

An *event* is something like a 'rain that begins', 'the sunshine happens', or a person decided to 'act' in some way, e.g. to 'open a door', to 'enter a number in a keypad' etc. Such events have usually some *effect* which changes properties. For the construction of the story this makes an interesting difference. To describe the *change-potential* of a state one can introduce a set of change-statements which only state, *which kinds of events can occur*. If they do not occur nothing will change, if they occur then a new state will be generated with this event as a new property. Is an event introduced, then this event can *succeed* by causing some *effect* or the event will not succeed; this will change nothing and the story returns to an earlier state before this event occurred.[3]

If an event is introduced in a new state and the event will *succeed* than the effects are described either by properties which will be *deleted* in the successive state or which will be *created newly*.

Thus we have two kinds of change-statements:

$$
\begin{array}{rcl}
\Xi & = & \{ \\
OCCUR\{ & & \{(Q \times Q') \times \{'occurs'\} \times d\{...\} \times c\{'CT'\}\}, ...\} \\
\neg OCCUR\{ & & \{(Q \times Q) \times \{'\neg occurs'\} \times d\{...\} \times c\{\}\}, ...\} \\
SUCCEED\{ & & \{(Q \times Q'^{+}) \times CT \times d\{...\} \times c\{...\}\}, ...\} \\
\neg SUCCEED\{ & & \{(Q \times Q'^{-}) \times \{'\bot CT'\} \times d\{...\} \times c\{\}\}, ...\} \\
& & \}
\end{array}
$$

The *occurrence* of events will introduce in the follow up states (Q') complex terms (CT) as names for the events. If they do not occur nothing will change. If an introduced event *succeeds* then some effect will change the follow-up state. If the event will *not succeed* ($\bot$) then nothing will change but the story leads back *before* that state where the event occurred (Q'^{-}). This allows a kind of *repetition*.

A practical device to realize such a mapping within an extended graph is the following procedure:

1. Given a vertex v with the properties $\{P_1, ..., P_n\}$ enhanced by some complex terms.

2. 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, $change = d \otimes c$.

3. The expression 'operation() : d{}, c{}' defines the meaning of the operation by the deletion and creation information.

4. The full change-expression is written as follows: $\langle v, v', operation - name() : d\{\}, c\{\}\rangle$. In case an event does not succeed one writes $\langle v, v', \bot operation - name() : d\{\}, c\{\}\rangle$

[3] This handling of complex terms as potential operations embedded in a state has a certain similarity with the view of quantum mechanics onto reality (cf. Feynman (1985) or Görnitz (2006)). A 'state' in quantum mechanics is always a space of possibilities with varying probabilities. To transform the possibilities in measurable (classical) units one has to reduce the possibilities. The concept of the actor story in this book looks to states as a structure which is enhanced with a set of (here 'finitely many') *possible actions* which can become real' but must not.

Richard Feynman. *QED. the strange theory of light and matter*. Princetn University Press, Princeton, Oxford, 1 edition, 1985; and Thomas Görnitz. *Quanten sind anders. Die verborgene Einheit der Welt*. Spektum Akademischer Verlag, Heidelberg (Germany), 1 edition, 2006

CHANGE STATEMENTS AND PROBABILITY: Until now a state can become associated with a finite set of change-statements indicating possible operations. Without further information it is nearly *impossible to decide* which one of these possible changes shall happen. Usually the problem will be solved by the *participating actors* which have *individual behavior functions* and therefore are the actors the *source of information* what will be done. If the actors will be *deterministic* systems then it can happen that repetitions of the actor story can reveal some stable frequencies of selections. If the actors are *not deterministic* then the outcome can vary and therefore the observed frequencies of selections can vary too.

4.3.3 Example MAS

In this paragraph it will be shown how one can construct a mathematical actor story (MAS). According to the formal definition of an AAI-expert generating a MAS (cf. 4.7) we have the following basic mapping:

$$\tau_{mth} \quad : \quad D^{Ref} \times L_{mth} \longmapsto MAS$$

This describes the case where an expert generates a MAS 'from scratch'. In this book it will be assumed that a textual actor story TAS will be generated first. Having such a textual version gives a better starting point because the construction of a TAS has realized a pre-selection of a subset of the domain of reference. The new construction of a mathematical actor story can use the 'pre-formatted' domain of reference using the TAS and the interpretation function τ_{txt}^{-1}. The format of the mapping function τ_{mth} has then the format:

$$\tau_{mth} \quad : \quad D^{Ref} \times TAS \times L_{mth} \longmapsto MAS$$

Thus we take as a starting point the simple TAS from above:

1. *Start*: A person A stands before a closed door. Besides the door there is a numeric keypad.

2. *Entering Key*: The person A enters a key into the keypad. GOTO state named 'Goal'. ELSE GOTO state named 'Start'.

3. *Goal*: The door is open.

In a first step one tries to identify at which point in the actor story at least one property is changing. These changes can then be used to identify a change as a succession of two states: the state before the change q and the state with the change q'. Good indicators of changes are *actions* or *events*. Such an action is given with the expression 'enters a key'. Thus we have a state $q1$ before this action and a state $q2$ with the action. And from this follows by definition that there must exist a follow-up state $q3$ to the action which represents the change, if there is a change because the action was *successful*. If the action was *not successful* then one has to decide what should be the case. If it should be possible that the action can be *repeated* then one has to *go back* to a state *before* the action can happen. In this case this could be state $q1$ again. Because there is no further change identifiable the actor story ends here; a very 'short story'...

From this analysis we can derive a fist rough version of the mathematical actor story as follows:

$$\begin{aligned} MAS_1(g1) \quad iff \quad & g1 = \langle V1, E1, \Pi1, \Xi1, \lambda1, \epsilon1 \rangle \\ V1 \quad = \quad & \{q1, q2, q3\} \\ E1 \quad = \quad & \{(q1, q2), (q2, q1), (q2, q3)\} \end{aligned} \tag{4.11}$$

What is still missing this are the properties of the different states and the details of the changes. First one can define the properties of each state:

1. *q1*: A person A stands before a closed door. Besides the door there is a numeric keypad.
2. *q2*: The person A enters a some key into the keypad.
3. *q3*: The door is open.

In state $q1$ there are three objects: a person, a door, and a keypad. These will be *named* as $U1, D1, Kp$. These names are associated with *attributes* like *being a person', 'being a door', 'being closed'*, and *'being a numeric keypad'*. These attributes will be encoded as $PERSON(U1)$, $DOOR(D1)$,$CLOSED(D1)$, and $KEYPAD(Kp)$. Then there are two *relations*: 'before' and 'besides'. This will be encoded as $BEFORE(U1, D1)$ and $BESIDES(Kp, D1)$. Thus the set of properties of state $q1$ can be summarized as follows:

1. $q1 = \{PERSON(U1), DOOR(D1), KEYPAD(Kp), CLOSED(D1), BEFORE(U1, D1), BESIDES(Kp, D1)\}$.
2. *q2*: The person A enters a some key into the keypad.
3. *q3*: The door is open.

In state $q2$ we can define again three objects: person, key, keypad. These will be *named* as $U1, K1, Kp$. These names are associated with *attributes* like *'being a person', 'being a numeric keypad', 'being a key'*. These attributes will be encoded as $PERSON(U1)$, $KEY(K1)$, and $KEYPAD(Kp)$. Then there is a special property called an *action* with the expression 'enters ...'. This action-expression is a complex term with three arguments: the actor doing the action (the person), some object used within this action (the key), and the goal of this action (the keypad). This will be encoded as $enter(U1, K1, Kp)$. This gives the following complete state $q2$. But attention: although the textual version of the actor story does not mention the other properties which are known from state $q1$ they are still in existence, if not otherwise told. And this can be done only by a change statement (see below).

1. $q1 = \{PERSON(U1), DOOR(D1), KEYPAD(Kp), CLOSED(D1), BEFORE(U1, D1), BESIDES(Kp, D1)\}$.

2. $q2 = \{PERSON(U1), KEY(K1), DOOR(D1), KEYPAD(Kp), CLOSED(D1), BEFORE(U1, D1), BESIDES(Kp, D1), enter(U1, K1, Kp)\}$

3. *q3*: The door is open.

In state $q3$ does the textual version only mention one object, the door. But clearly the other objects 'person and keypad did not vanish. Therefore we have still the objects $U1, D1, Kp$. While the attributes for the object person and keypad did not change, it is now told, that die property of the door has change; it is now 'open'. Therefore we get the following list of properties: $PERSON(U1)$, $KEYPAD(Kp), DOOR(U1), OPEN(D1)$. Furthermore we have still the two *relations*: 'before' and 'besides'. Thus the set of properties of state $q3$ have to be summarized as follows:

1. $q1 = \{PERSON(U1), DOOR(D1), KEYPAD(Kp), CLOSED(D1), BEFORE(U1, D1), BESIDES(Kp, D1)\}$.

2. $q2 = \{PERSON(U1), KEY(K1), DOOR(D1), KEYPAD(Kp), CLOSED(D1), BEFORE(U1, D1), BESIDES(Kp, D1), enter(U1, K1, Kp)\}$

3. $q3 = PERSON(U1), DOOR(D1), OPEN(D1), KEYPAD(Kp), BEFORE(U1, D1), BESIDES(Kp, D1)$.

Now we have done the following construction:

$$\begin{array}{rcl} MAS_1(g1) & iff & g1 = \langle V1, E1, \Pi1, \Xi1, \lambda1, \epsilon1 \rangle \\ V1 & = & \{q1, q2, q3\} \\ E1 & = & \{(q1, q2), (q2, q1), (q2, q3)\} \\ \Pi1 & = & q1 \cup q2 \cup q3 \\ \lambda1 & : & q1 = \{\}, q2 = \{\}, q3 = \{\} \end{array} \tag{4.12}$$

What is still missing this are the change statements explaining which kinds of change happened. As one can see there are two changes: (i) the *occurrence* of an action in $q2$ and (ii) the *effect of the action* in $q3$ if the action *succeeds*. According to the definition of the change statement 4.11 we have then to construct change-expressions like this:

$$\begin{array}{lcl} \Xi1 & = & \{ \\ \{ \quad OCCURS & & \langle (q1, q2), act(U1), d\{\}, c\{enter(U1, K1, Kp)\}\rangle \\ \neg OCCURS & & \langle (q1, q1), \neg act(U1), d\{\}, c\{\}\rangle\} \\ \{ \quad SUCCEEDS & & \langle (q2, q3), enter(U1, K1, Kp), d\{CLOSED(D1)\}, c\{OPEN(d1)\}\rangle \\ \neg SUCCEEDS & & \langle (q2, q1), \perp enter(U1, K1, Kp), d\{\}, c\{\}\rangle\} \\ & & \} \end{array} \tag{4.13}$$

From this we can complete the mathematical actor story as follows:

1. $q1 = \{PERSON(U1), DOOR(D1), KEYPAD(Kp), CLOSED(D1), BEFORE(U1, D1), BESIDES(Kp, D1)\}$.

2. $\{\langle (q1, q2), act(U1), d\{\}, c\{enter(U1, K1, Kp)\}\rangle, \langle (q1, q1), \neg act(U1), d\{\}, c\{\}\rangle\}$

3. $q2 = \{PERSON(U1), KEY(K1), DOOR(D1), KEYPAD(Kp), CLOSED(D1), BEFORE(U1, D1), BESIDES(Kp, D1), enter(U1, K1, Kp)\}$

4. $\{\langle (q2, q3), enter(U1, K1, Kp), d\{CLOSED(D1)\}, c\{OPEN(d1)\}\rangle, \langle (q2, q1), \perp enter(U1, K1, Kp), d\{\}, c\{\}\rangle\}$

5. $q3 = PERSON(U1), DOOR(D1), OPEN(D1), KEYPAD(Kp), BEFORE(U1, D1), BESIDES(Kp, D1).$

4.3.4 Picture-to-Math Mapping

If one has already a picture language at hand enabling pictorial actor stories then one can also establish a *picture-to-math mapping (P2M)* analogously to the picture-to-text mapping.. A simple example is demonstrated below (cf. figure 4.3):

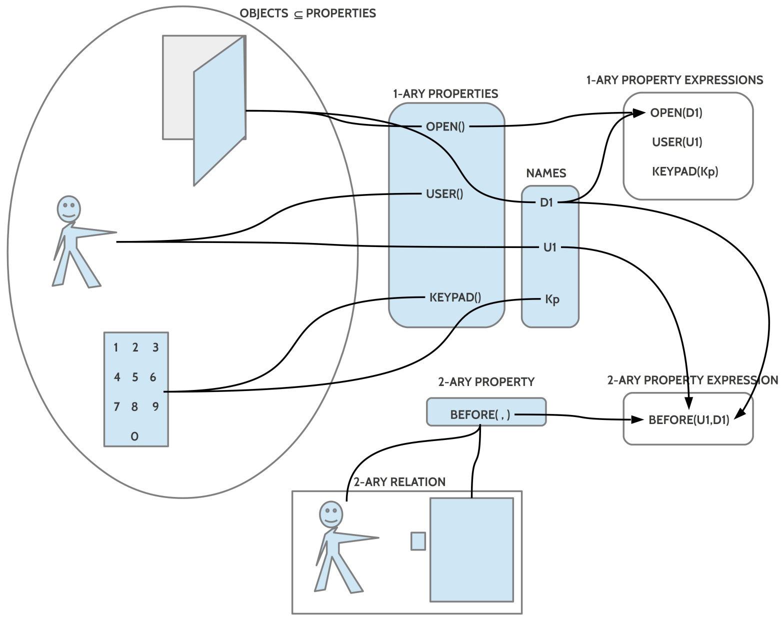

Figure 4.3: Example of a picture-to-math mapping for properties and objects

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()'} are talking about features/ attributes of an object/ actor, and 2-ary properties are talking about relations and 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'

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

4.4 Task Induced Actor Requirements (TAR)

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

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

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

4.5 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, $\sigma : RProf_{requ} \times RProf_{avail} \longmapsto [0,1]$.

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

4.6 Interface-Requirements and Interface-Design

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

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

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

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

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

5
Actor Model (AM)

5.1 Actor Model and Actor Story

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

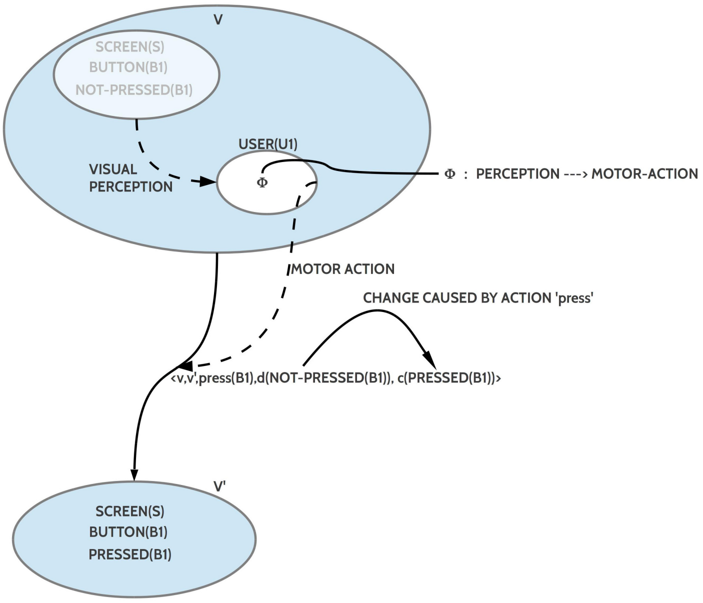

Figure 5.1: Change event with an embedded actor

The general idea of this interaction between actor story and actor model can be seen in figure 5.1.

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' Φ the actor can compute a possible output as a motor-action, described as an event expression $\langle v, v', press(BUTTON(B1)), d(not - pressed(B1)), C : (pressed(B1))\rangle$.

3. This results in a change leading to v'. The actor $U1$ is left out in v', also it is still part of v'.

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 pairs $\{(i,o),\cdots,(i,o)\}$ defining implicitly a required *empirical behavior function*:

$$\phi_e = \{(i,o),\cdots,(i,o)\} \tag{5.1}$$

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:

$$\phi : I \longmapsto O \tag{5.2}$$

Although an empirical behavior function ϕ_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* ϕ. 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* ϕ_e, then the hypothetical model is *empirical sound in a weak sense*.

$$\phi_e \subseteq \phi \tag{5.3}$$

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'.

5.2 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 σ 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 σ_A. Thus we have a chain $\sigma_E : 2^{\Pi} \longmapsto ACT$ and then $\sigma_A : rn(\sigma_E) \longmapsto I_A$. 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 μ_A into some external properties of the actor A, which in turn are then translated by the response function of the environment μ into some effects represented as deletion of existing properties $2^{\Pi-}$ as well as of creation of new properties $2^{\Pi+}$: $\mu_A : O_A \longmapsto O_{A,resp}$ and then $\mu : rn(\mu_A) \longmapsto 2^{\Pi-} \cup 2^{\Pi+}$.

Thus we get a *hierarchical embedding* of structures:

$$\begin{aligned}
ENV(E) &:= Environment\ E \\
ENV(E) &\ iff\ E = \langle \Pi, ACT, \sigma, \mu \rangle \\
\Pi &:= Set\ of\ properties \\
ACT &:= Set\ of\ actors \\
ACT &\subseteq 2^{\Pi} \\
\sigma &: 2^{\Pi} \longmapsto ACT(stimulus\ function) \\
\mu &: O_{ACT,resp} \longmapsto 2^{\Pi}(response\ function)
\end{aligned} \tag{5.4}$$

and:

$$\begin{aligned}
ACT(A) &:= Actor\ A \\
ACT(A) &\ iff\ A \in ACT \wedge A = \langle I, O, IS, \sigma, \mu \rangle \\
I_A &:= Input \\
O_A &:= Output
\end{aligned} \tag{5.5}$$

$$\begin{aligned} \sigma_A &: rn(\sigma_E) \longmapsto I_A \\ \mu_A &: O_A \longmapsto O_{A,resp} \end{aligned}$$

and:

$$\begin{aligned} IOSYS(S) &:= Input - Output\ System \\ IOSYS(S) &\ iff\ S = \langle I, O, IS, \phi \rangle \\ I &:= Input \\ O &:= Output \\ IS &:= Internal\ States(can\ be\ empty) \\ \phi &: I \times 2^{\Pi} \times 2^{\Pi} \times O \end{aligned} \tag{5.6}$$

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]

[1] Here is the environment defined by the actor story.

With these clarifications it becomes clear that the 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 $rn(\sigma_E)$ as well as the response-values of the actor $O_{A,resp}$. Thus we have:

$$BIntf_{A,E} = \{x | x \in rn(\sigma_E) \times O_{A,resp}\}$$

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 (BInf)* 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 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'.

5.3 Input-Output Systems Basic Typology

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 5.2).

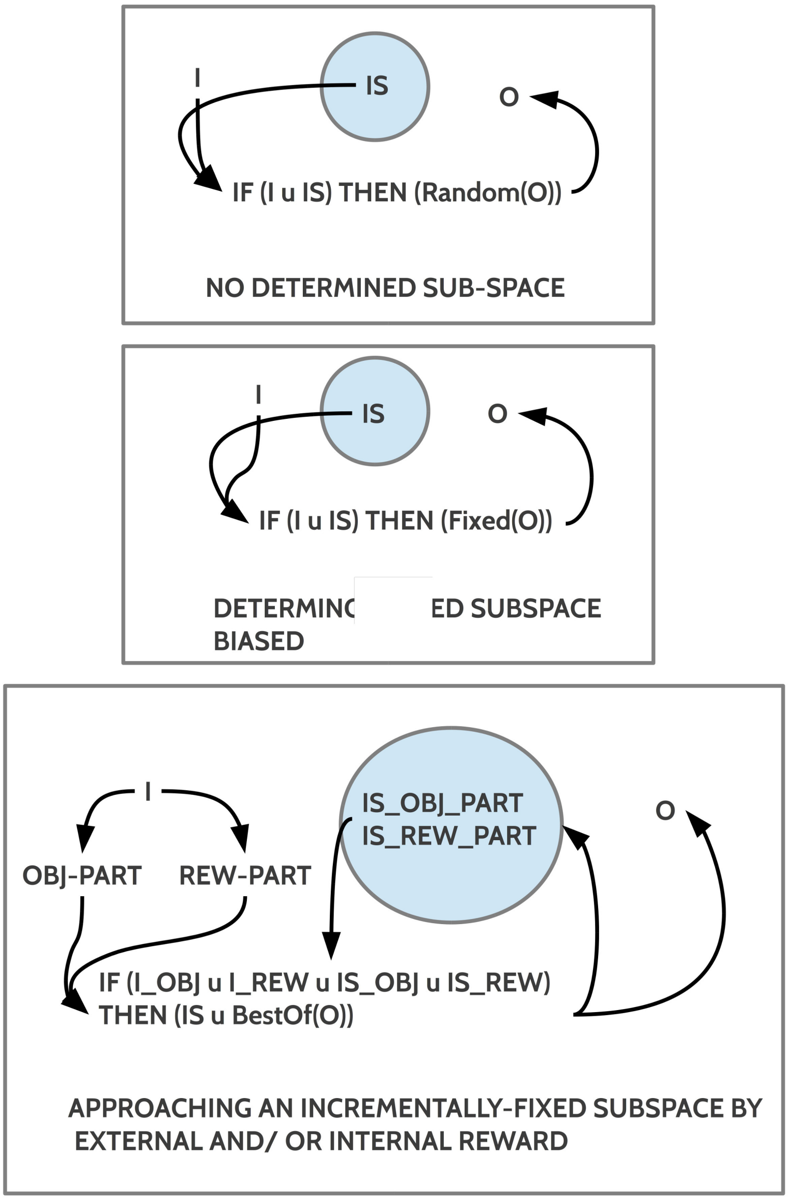

Figure 5.2: Basic Typology of Input-Output Systems

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 $O_{Fixed,SYS}$ 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 $O_{Sel,SYS}$ depending on some kinds of *rewards* which can be part either of the *external* input I or of some *internal* states IS_{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 $O_{Random,SYS}, O_{Fixed,SYS}, O_{Sel,SYS}$ then one can compare these special sets with the *general* set of system-dependent outputs O_{SYS} and the set of *possible* outputs offered by the actor story as the *world (W)* given as O_W. If one takes the possible outputs of the world called O_W as point of reference then the system dependent outputs $O_{Random,SYS}, O_{Fixed,SYS}, O_{Sel,SYS}, O_{SYS}$ are usually *true subsets* of the possible world outputs and there can be intriguing overlaps between $O_{Random,SYS}, O_{Fixed,SYS}, O_{Sel,SYS}$. There can be cases that the learning system with its output set $O_{Sel,SYS}$ is weaker then the system with a fixed output set $O_{Fixed,SYS}$ and this in turn can be weaker than a random system with the random output set $O_{Random,SYS}$. 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 (BIntf)' 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.

5.4 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.

5.4.1 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:

[2] Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc., Mahwah (NJ), 1 edition, 1983

$$\phi = \phi_{perc} \otimes \phi_{cogn} \otimes \phi_{mot} \quad (5.7)$$
$$\phi_{perc} := Perception \quad (5.8)$$
$$\phi_{perc} : I \longmapsto (VB \cup AB) \quad (5.9)$$
$$VB := Visual\ buffer \quad (5.10)$$
$$AB := Auditory\ buffer \quad (5.11)$$
$$\phi_{cogn1} : (VB \cup AB) \times M_{STM} \longrightarrow M_{STM} \quad (5.12)$$
$$\phi_{cogn2} : M_{STM} \times M_{LTM} \longrightarrow M_{STM} \times M_{LTM} \quad (5.13)$$
$$\phi_{cogn1+2} := Cognition \quad (5.14)$$
$$\phi_{mot} : M_{LTM} \longrightarrow O \quad (5.15)$$
$$\phi_{mot} := Motor\ activity \quad (5.16)$$

Thus an input – visual or auditory – will be processed by the *perception function* ϕ_{perc} into an appropriate *sensory buffer* VB oder AB. The contents of the sensory buffers will then be processed by the partial *cognitive function* $cogn_1$ into the *short term memory (STM)*, which at the same time can give some input for this processing. Another cognitive function $cogn_2$ 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:

$$VB \cup AB \cup M_{STM} \cup M_{LTM} \subseteq IS \quad (5.17)$$

The complete model can be found in the cited book.

5.4.2 Non-Empirically Motivated

In many cases non-empirically motivated models are sufficient. This amounts to the task to 'invent' a function ϕ 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.

5.4.3 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 **??** a rule of a GOMS model has the logical format:

$$IF\ I = X \wedge IS = Y \quad THEN \quad IS = Y' \wedge O = Z \tag{5.18}$$

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 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.

[3] A first extensive usage of a GOMS model can be found in Card et al. (1983) :139ff

Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc., Mahwah (NJ), 1 edition, 1983

[4] For a description of the example see: http://www.doeben-henisch.de/fh/fsv/node13.html in Doeben-Henisch (2010) .

Gerd Doeben-Henisch. *Formal Specification and Verification: Short Introduction*. Gerd Doeben-Henisch, 2010

1. Start = U1 $\cup$ S1 $\cup$ Env1
   U1 = {USER(U1)}
   S1 = {SYSTEMINTF(S1),KEYPAD(K1), PART-OF(K1, S1), KEY(Ka), KEY(Kb),KEY(Kc),PART-OF($\langle Ka, Kb, Kc\rangle$, 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:$\langle$ Start,Start,push(not(Ka),K1),d(),c()$\rangle$, $\langle$ Start,Q2,push(Ka,K1), d(), c(PRESSED(Ka))$\rangle$,

3. Q2 = U1 $\cup$ S1 $\cup$ Env1
   U1 = {USER(U1)}
   S1 = {SYSTEMINTF(S1),KEYPAD(K1), PART-OF(K1, S1), KEY(Ka), KEY(Kb),KEY(Kc),PART-OF($\langle Ka, Kb, Kc\rangle$, K1),PRESSED(Ka)}
   Env1 = {DOOR(D1), CLOSED(D1)}

4. CHANGE-AS: $\langle Q2, Start, push(not(Kb), K1), d(), c()\rangle$, $\langle$ Q2,Q3,push(Kb,K1), d(), c(PRESSED(Kb))$\rangle$,

5. Q3 = U1 $\cup$ S1 $\cup$ Env1
   U1 = {USER(U1)}
   S1 = {SYSTEMINTF(S1),KEYPAD(K1), PART-OF(K1, S1), KEY(Ka), KEY(Kb),KEY(Kc),PART-OF($\langle Ka, Kb, Kc\rangle$, K1),PRESSED(Kb)}
   Env1 = {DOOR(D1), CLOSED(D1)}

6. CHANGE-AS: $\langle Q3, Start, push(not(Ka), K1), d(), c()\rangle$, $\langle$ Q3,Goal,push(Ka,K1), d(CLOSED(D1)), c(PRESSED(Ka), OPEN(D1))$\rangle$

7. Goal = U1 $\cup$ S1 $\cup$ Env1
   U1 = {USER(U1)}
   S1 = {SYSTEMINTF(S1),KEYPAD(K1), PART-OF(K1, S1), KEY(Ka), KEY(Kb),KEY(Kc),PART-OF($\langle Ka, Kb, Kc\rangle$, 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 5.7, should be preferred: it gives implicitly all necessary informations and keeps the amount of written information low.

[5] Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gem-Phong Vo. A technique for drawing directed graphs. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 19(3):214–230, 1993

[6] Emden R. Gansner, Yehuda Koren, and Stephen North. Graph drawing by stress majorization. In János Pach, editor, *Graph Drawing*, number 3383 in Lecture Notes in Computer Science, pages 239 – 250, Berlin - Heidelberg. Springer-Verlag

[7] Emden R. Gansner, Eleftherios Koutsofios, and Stephen C. North. Drawing graphs with dot. pages 1–40, 2015. Online: http://www.graphviz.org/pdf/dotguide.pdf

5.4.4 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

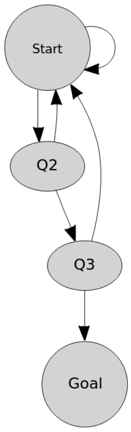

Figure 5.3: Electronic door example - bare graph, only nodes

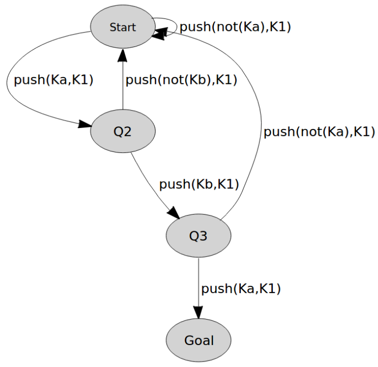

Figure 5.4: aai-example electronic door: nodes and minimally labeled edges

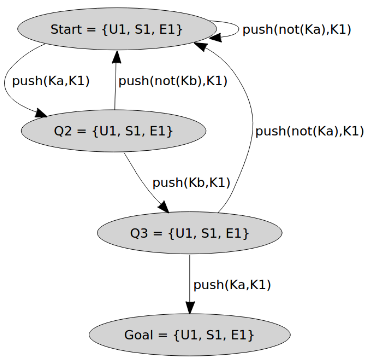

Figure 5.5: aai example electronic door with nodes, shortened edge-labels, and subsets of properties

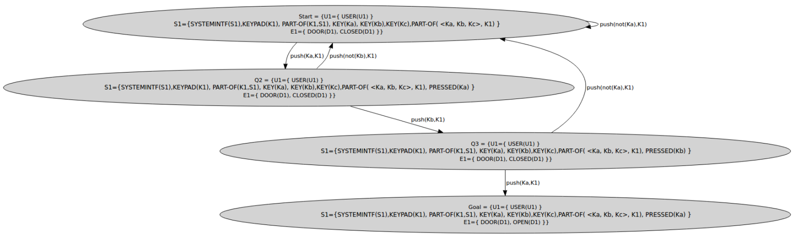

Figure 5.6: aai example with a complete graph (only the edge labels are shortened)

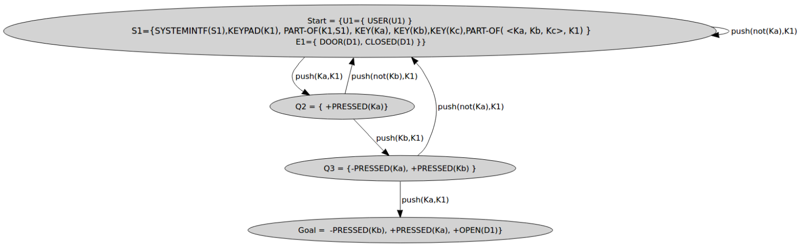

Figure 5.7: Graph with complete start state followed by difference states based on labelled edges

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 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.

[7] Instead of using the GOMS format for an actor model one can use every kind of a function, e.g. a function ϕ realized with a normal programming language like 'C/C++', 'Java', 'python' etc.

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 $\phi : I \times 2^{IS} \longmapsto 2^{IS} \times O$

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) IF K1 ='KEY-PRESSED(K1,not(Kb))' & IS='CODE(C2,(Ka,1,Kb,0,Ka,0))'
THEN IS='CODE(C2,(Ka,0,Kb,0,Ka,0))'

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

(f) IF K1 ='KEY-PRESSED(K1,not(Ka))' & IS='CODE(C2,(Ka,1,Kb,1,Ka,0))'
THEN IS='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.

5.5 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. Eysenk (2004) [8], Rost (2009) [9], Rost (2013) [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) [11] and a famous experiment with Tolman (1948)[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) [13], English translation [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 ϕ represent nothing else as a

[8] Hans J. Eysenk. *Die IQ-Bibel. Intelligenz verstehen und messen*. J.G.Cotta'sche Buchhandlung Nachfolger GmbH, Stuttgart, 1 edition, 2004. Englische Originalausgabe 1998: Intelligence. A New Look

[9] Detlef H. Rost. *Intelligenz. Fakten und Mythen*. Beltz Verlag, Weinheim - Basel, 1 edition, 2009

[10] Detlef H. Rost. *Handbuch Intelligenz*. Beltz Verlag, Weinheim - Basel, 1 edition, 2013

[11] Ernest R. Hilgard, Rital L. Atkinson, and Richard C. Atkinson. *Introduction to Psychology*. Harcourt Brace Jovanovich, Inc., New York - San Diego - Chicago - et.al., 7 edition, 1979

[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

[13] Hermann Ebbinghaus. *Ű̈ber das Gedắchtnis: Untersuchungen zur experimentellen Psychologie*. Duncker & Humblot, Leipzig, 1 edition, 1885. URL: http://psychclassics.yorku.ca/Tolman/Maps/maps.htm

[14] Hermann Ebbinghaus. *Memory: A Contribution to Experimental Psychology*. Teachers College, Columbia University, New York, 1 edition, 1913. Translated from the German Edition 1885 by Henry A. Ruger & Clara E. Bussenius 1913, URL: http://psychclassics.yorku.ca/Ebbinghaus/index.htm

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

6
Simulation

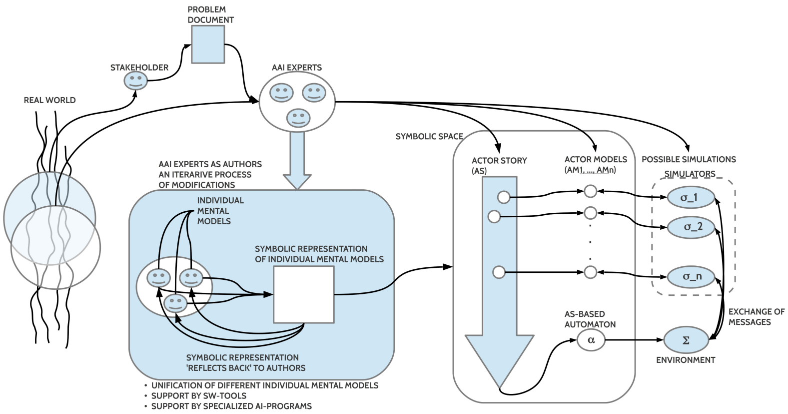

Figure 6.1: Creation of the symbolic space by AAI-experts including an actor story (AS), actor models (AMs), automata based on the actor story as well as on the actor models, and simulators running simulations with the automata.

GENERAL OUTLINE: Figure 10.2 outlines the symbolic space which is basically constituted by an actor story in mathematical mode (AS_{ϵ}) and – optionally – by several actor models primarily also in mathematical mode (AM_{ϵ}). In both cases one can *compile* these representations into equivalent *algorithmic representations* which represent *automata* M_{α}. 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_{σ}.

While the simulation model of the actor story $M_{\sigma,AS}$ written as M_{Σ} 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_{σ,AM_i} written as M_{σ_i} individual actors operating in the framework of the actor story.

ALGORITHMIC CONVERSION: A given *extended graph* γ^{+} can be mapped into an *automaton* $M_{\alpha,AM}$ by a direct mapping. As starting point we take an *extended ordered graph (EOG)* as follows:[1]

$$\begin{aligned} EOG(x) \quad iff \quad & x = \langle V, I, F, \Pi, \chi, E, \lambda, \chi \rangle \qquad (6.1) \\ V \quad := \quad & finite\ set\ of\ vertices \end{aligned}$$

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

H.L. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publ.Company, Reading (MA), 1 edition, 1979

$$\begin{aligned}
I &\subseteq V; set\ of\ initial\ vertices \\
F &\subseteq V; set\ of\ final\ vertices \\
\Pi &:= finite\ set\ of\ property\ expressions \\
\chi &:= finite\ set\ of\ action\ expressions \\
E &\subseteq V \times V; set\ of\ edges \\
\lambda &: V \longmapsto 2^{\Pi} \\
\chi &: E \longmapsto \chi
\end{aligned} \tag{6.2}$$

The usual definition of a finite automaton is as follows:

$$\begin{aligned}
FA(x) \quad iff &\quad x = \langle Q, I, F, \Sigma, \Delta \rangle \quad (6.3) \\
Q &:= finite\ set\ of\ states \\
I &\subseteq Q; set\ of\ initial\ states \\
F &\subseteq F; set\ of\ final\ states \\
\Sigma^* &:= input\ words \\
\Delta &\subseteq Q \times \Sigma^* \times Q; set\ of\ transitions
\end{aligned}$$

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 Π as follows: $\langle Q, I, F, \Sigma, \Pi, \Delta \rangle$ and with $\lambda : Q \longrightarrow 2^{\Pi}$, 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 Π. All state-transitions of the automaton α^+ from a start-state to a goal-state together are called a *run* ρ of the automaton. The set of all possible runs of the automaton is called the *execution graph* γ_{exec} of the automaton α^+ or $\gamma_{exec}(\alpha^+)$.

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

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.

7 Algorithmic Verification

Another helpful process is the process named ν. It translates a behavior model M_{SR} with the aid of a *temporal logic language* L_{TL} and an appropriate algorithm α into a *algorithmic verification model* M_{ν}, 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 hove to demonstrate here one new approach to overcome the known problems.

The following selected papers (only a subset of thematic related papers) can illustrate the discussion until now. [1].

[1] F. Khalique, W. H. Butt, and S. A. Khan. Creating domain non-functional requirements software product line engineering using model transformations. In *2017 International Conference on Frontiers of Information Technology (FIT)*, pages 41–45, Dec 2017. DOI: 10.1109/FIT.2017.00015; F. Fellir, K. Nafil, and R. Touahni. Analyzing the non-functional requirements to improve accuracy of software effort estimation through case based reasoning. In *2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*, pages 1–6, Oct 2015. DOI: 10.1109/SITA.2015.7358402; D. Mairiza, D. Zowghi, and V. Gervasi. Conflict characterization and analysis of non functional requirements: An experimental approach. In *2013 IEEE 12th International Conference on Intelligent Software Methodologies, Tools and Techniques (SoMeT)*, pages 83–91, Sept 2013. DOI: 10.1109/SoMeT.2013.6645645; A. Suhr, C. Rosinger, and H. Honecker. System design and architecture ?? essential functional requirements vs. ict security in the energy domain. In *International ETG-Congress 2013; Symposium 1: Security in Critical Infrastructures Today*, pages 1–9, Nov 2013; B. Yin, Z. Jin, W. Zhang, H. Zhao, and B. Wei. Finding optimal solution for satisficing non-functional requirements via 0-1 programming. In *2013 IEEE 37th Annual Computer Software and Applications Conference*, pages 415–424, July 2013. DOI: 10.1109/COMPSAC.2013.69; X. L. Zhang, C. H. Chi, C. Ding, and R. K. Wong. Non-functional requirement analysis and recommendation for software services. In *2013 IEEE 20th International Conference on Web Services*, pages 555–562, June 2013. DOI: 10.1109/ICWS.2013.80; ; Y. Liu, Z. Ma, R. Qiu, H. Chen, and W. Shao. An approach to integrating non-functional requirements into uml design models based on nfr-specific patterns. In *2012 12th International Conference on Quality Software*, pages 132–135, Aug 2012. DOI: 10.1109/QSIC.2012.23; and M. Kassab, O. Ormandjieva, and M. Daneva. An ontology based approach to non-functional requirements conceptualization. In *2009 Fourth International Conference on Software Engineering Advances*, pages 299–308, Sept 2009. DOI: 10.1109/ICSEA.2009.50

8
Physical Design

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.

9
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. Therefor one has to realize a series of tests to check the quality of the observable real processes compared to the logical requirement of the actor story.

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

To find the *relative best* demonstrator in a finite set of candidate demonstrators $\{(D_1, V_1), (D_2, V_2), ..., (D_m, V_m)\}$ one has to define a *measure* $\mu : 2^{V^n} \longmapsto V^n$ for the assumed finite many n-dimensional values $\{V_1^n, V_2^n, ..., V_m^n\}$ to compare these values and identify for this set an *optimal value.* Thus $\mu(V_1^n, V_2^n, ..., V_m^n)$ computes a certain $V_i^n \in \{V_1^n, V_2^n, ..., V_m^n\}$.

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

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

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

1. Receiving an *actor story AS* and the *TAR document* from that story.
2. Selecting a group of *test candidates* $\{T_1, ..., T_n\}$ planned to mimicking the intended executive actor.
3. *Work out an AAR document* for *each* of the test candidates yielding a set of pairs $\{(T_1, AAR_1), ..., T_n, AAR_n)\}$.
4. Compute the *distance* of each AAR_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 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_{ϵ}, 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.

[1] Which is a highly idealistic assumption in case of learning systems

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 th 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_ϵ 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.

10.1 The Logical Space

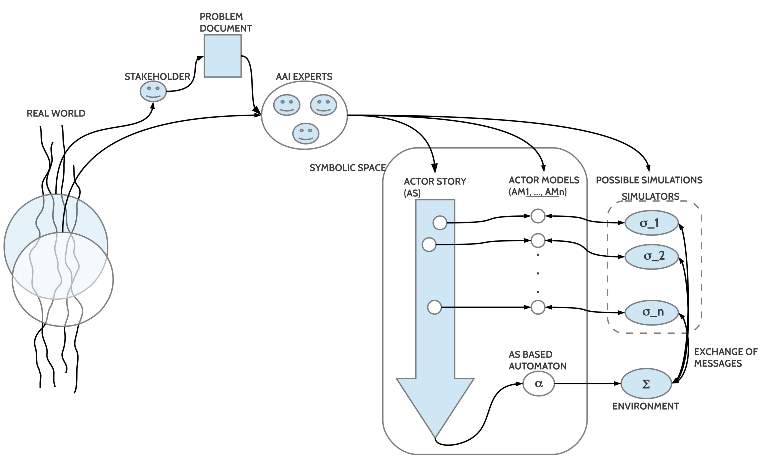

Figure 10.1: The actor story (AS) and the actor models (AMs) as symbolic representations constituting a symbolic space

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 Σ simulating the actor story is representing the actor story is the *world* of the simulation, and the different simulators $\sigma_1, ..., \sigma_n$ 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.

10.2 Creating the Symbolic Space

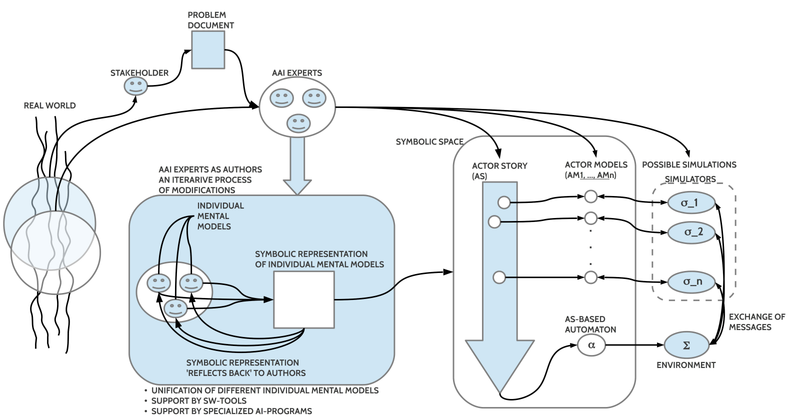

Figure 10.2: Creation of the symbolic space by AAI-experts

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 kown world.(See for this [2] and

[2] Gerd Doeben-Henisch. Philosophy of the actor. *eJournal uffmm.org*, pages 1–8, 2018. ISSN 2567-6458. URL https://www.uffmm.org/2018/03/20/actor-actor-interaction-philosophy-of-the-actor/

figure **??**) 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.

10.3 The Physical Space

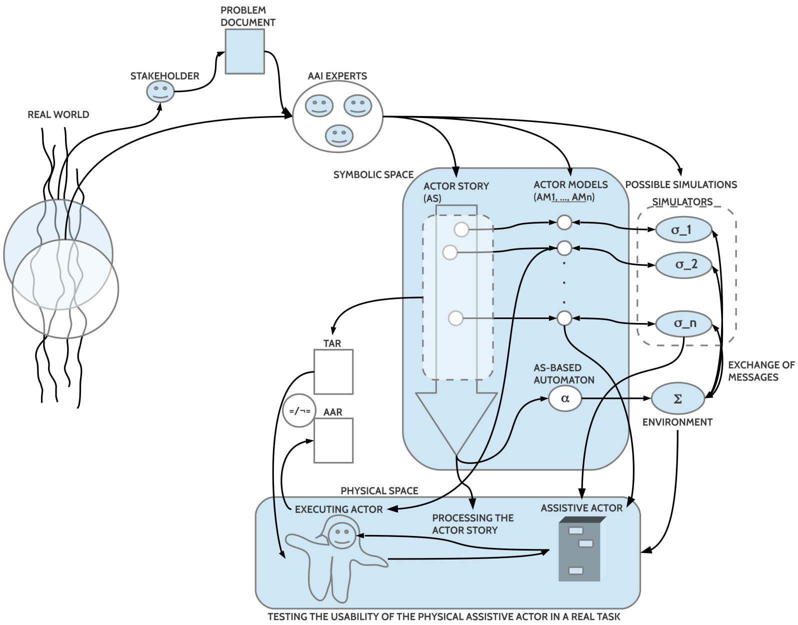

Figure 10.3: The symbolic space extended with simulators translated into the physical space with physical actors as well as physical assistive actors

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.

10.4 Multiple Actor Stories

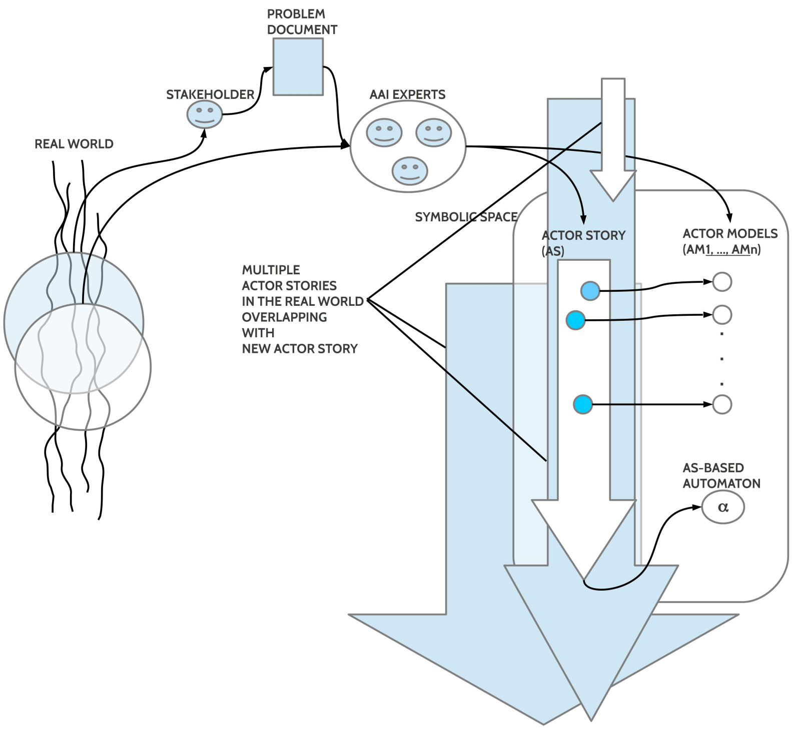

Figure 10.4: The intended actors of an actor story (AS) are living as real actors usually in more than one actor story

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.[1]

[1] 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) [2], (2018)[3].

[2] 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

[3] 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/

Part II

Application

12
Applying a Theory

While the *part I* of this book introduces the main concepts and terms from a theoretical point of view, shall *part II* provide the description of a possible real process: how to apply the theoretical concepts in a way that one can get as a result a real actor story with real models and a real simulation with real usability tests. It is left to *part III* to give a first experimental example of a software which shall support the real process.

Part III

Software

13
Software Tool: Introduction

To apply the theory described in this book in a real process one needs a *supporting software*.

It is assumed that the software tool is available through a website either as a *free download version* for local use or as a *web-application*.

1. In the beginning a textual actor story editor *TAS-editor* allows the editing of a textual actor story D_{tas} by the AAI experts as well as the stakeholder.

2. In the next step the textual version of the actor story D_{tas} will be translated in a *mathematical version* D_{mas} realized as an *extended graph* by a math-editor *MAS-editor*. This can be done at least in a semi-automatic mode.

3. Based on the D_{tas} and the D_{mas} one can edit with the text-to-math editor (*TMX-editor*) an adaptive TMX-lexicon D_{tmx} which list all important expressions of the text expressions associated with intended mathematical expressions.

4. Based on the textual D_{tas} and the mathematical D_{mas} version of the actor story one can edit a pictorial version of the actor story D_{pas} together with an adaptive text-to-math-to-diagram lexicon D_{tmpx}.

5. Already with these ingredients it is possible to generate in an *automatic mode* a simulator model σ_{TMP} which can simulate the actor story specified so far by showing a graphic interpretation. This can enhanced by several kinds of statistics as well as by an *interactive mode*.

6. Finally one can edit additionally *actor models* for the *assistive* actors as μ_{ass} or for the *executive* actors as μ_{exec}. This allows the inclusion of the actor behavior within the simulation mode. For this one has to extend the base simulator σ_{TMP} into a multi-model simulator $\sigma_{TMP.n}$ to enable real multi-model interactions. These simulations can be used for *training*, *testing* or for *gaming*. 'Gaming' is in this context not only 'fun' but serving research as an intensive way of *exploratory investigations*.

7. In a last step one can integrate in these simulators the *automatic verification tool* described in the theory part. This enhances the power of this software even more.

Bibliography

Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc., Mahwah (NJ), 1 edition, 1983.

G. Doeben-Henisch and M. Wagner. Validation within safety critical systems engineering from a computational semiotics point of view. *Proceedings of the IEEE Africon2007 Conference*, pages Pages: 1 – 7, 2007. DOI: 10.1109/AFRICON.2007.4401588.

Gerd Doeben-Henisch. *Formal Specification and Verification: Short Introduction*. Gerd Doeben-Henisch, 2010.

Gerd Doeben-Henisch. Philosophy of the actor. *eJournal uffmm.org*, pages 1–8, 2018. ISSN 2567-6458. URL https://www.uffmm.org/2018/03/20/actor-actor-interaction-philosophy-of-the-actor/.

Hermann Ebbinghaus. *Űber das Gedắchtnis: Untersuchungen zur experimentellen Psychologie*. Duncker & Humblot, Leipzig, 1 edition, 1885. URL: http://psychclassics.yorku.ca/Tolman/Maps/maps.htm.

Hermann Ebbinghaus. *Memory: A Contribution to Experimental Psychology*. Teachers College, Columbia University, New York, 1 edition, 1913. Translated from the German Edition 1885 by Henry A. Ruger & Clara E. Bussenius 1913, URL: http://psychclassics.yorku.ca/Ebbinghaus/index.htm.

Louwrence Erasmus and Gerd Doeben-Henisch. A theory of the system engineering process. In *9th IEEE AFRICON Conference*. IEEE, 2011a.

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.

Hans J. Eysenk. *Die IQ-Bibel. Intelligenz verstehen und messen*. J.G.Cotta'sche Buchhandlung Nachfolger GmbH, Stuttgart, 1 edition, 2004. Englische Originalausgabe 1998: Intelligence. A New Look.

F. Fellir, K. Nafil, and R. Touahni. Analyzing the non-functional requirements to improve accuracy of software effort estimation through case based reasoning. In *2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*, pages 1–6, Oct 2015. DOI: 10.1109/SITA.2015.7358402.

Richard Feynman. *QED. the strange theory of light and matter*. Princetn University Press, Princeton, Oxford, 1 edition, 1985.

Emden R. Gansner, Yehuda Koren, and Stephen North. Graph drawing by stress majorization. In János Pach, editor, *Graph Drawing*, number 3383 in Lecture Notes in Computer Science, pages 239 – 250, Berlin - Heidelberg. Springer-Verlag.

Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gem-Phong Vo. A technique for drawing directed graphs. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 19(3):214–230, 1993.

Emden R. Gansner, Eleftherios Koutsofios, and Stephen C. North. Drawing graphs with dot. pages 1–40, 2015. Online: http://www.graphviz.org/pdf/dotguide.pdf.

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.

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.

Kurt Goedel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, i. *Monatshefte fuer Mathematik und Physik*, 38: 173–98, 1931.

Thomas Görnitz. *Quanten sind anders. Die verborgene Einheit der Welt*. Spektum Akademischer Verlag, Heidelberg (Germany), 1 edition, 2006.

Jonathan Grudin. A Moving Target: The Evolution of HCI. In A. Sears and J.A. Jacko, editors, *The Human-Computer Interaction Handbook. Fundamentals, Evolving Technologies, and emerging Applications*. 2 edition, 2008.

Stephen Hawking. Gödel and the end of physics, 2002. URL http://www.hawking.org.uk/godel-and-the-end-of-physics.html.

Ernest R. Hilgard, Rital L. Atkinson, and Richard C. Atkinson. *Introduction to Psychology*. Harcourt Brace Jovanovich, Inc., New York - San Diego - Chicago - et.al., 7 edition, 1979.

H.L. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publ.Company, Reading (MA), 1 edition, 1979.

INCOSE. *SYSTEMS ENGINEERING HANDBOOK. A GUIDE FOR SYSTEM LIFE CYCLE PROCESSES AND ACTIVITIES*. John Wiley & Sons, Inc, Hoboken, New Jersey, 4 edition, 2015.

M. Kassab, O. Ormandjieva, and M. Daneva. An ontology based approach to non-functional requirements conceptualization. In *2009 Fourth International Conference on Software Engineering Advances*, pages 299–308, Sept 2009. DOI: 10.1109/ICSEA.2009.50.

F. Khalique, W. H. Butt, and S. A. Khan. Creating domain non-functional requirements software product line engineering using model transformations. In *2017 International Conference on Frontiers of Information Technology (FIT)*, pages 41–45, Dec 2017. DOI: 10.1109/FIT.2017.00015.

Y. Liu, Z. Ma, R. Qiu, H. Chen, and W. Shao. An approach to integrating non-functional requirements into uml design models based on nfr-specific patterns. In *2012 12th International Conference on Quality Software*, pages 132–135, Aug 2012. DOI: 10.1109/QSIC.2012.23.

D. Mairiza, D. Zowghi, and V. Gervasi. Conflict characterization and analysis of non functional requirements: An experimental approach. In *2013 IEEE 12th International Conference on Intelligent Software Methodologies, Tools and Techniques (SoMeT)*, pages 83–91, Sept 2013. DOI: 10.1109/SoMeT.2013.6645645.

Nils J. Nilsson, editor. *Artificial Intelligence*. Morgan Kaufmann, Menlo Park, 1998.

Richard W. Pew. Introduction. Evolution of human-computer interaction: From memex to bluetooth and beyond. In J.A. Jacko and A. Sears, editors, *The Human-Computer Interaction Handbook. Fundamentals, Evolving Technologies, and emerging Applications*. 1 edition, 2003.

Detlef H. Rost. *Intelligenz. Fakten und Mythen*. Beltz Verlag, Weinheim - Basel, 1 edition, 2009.

Detlef H. Rost. *Handbuch Intelligenz*. Beltz Verlag, Weinheim - Basel, 1 edition, 2013.

Stuart Russel and Peter Norvig. *Artificial Intelligence. A Modern Approach*. Universe Books, 3 edition, 2010.

A. Suhr, C. Rosinger, and H. Honecker. System design and architecture ?? essential functional requirements vs. ict security in the energy domain. In *International ETG-Congress 2013; Symposium 1: Security in Critical Infrastructures Today*, pages 1–9, Nov 2013.

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.

B. Yin, Z. Jin, W. Zhang, H. Zhao, and B. Wei. Finding optimal solution for satisficing non-functional requirements via 0-1 programming. In *2013 IEEE 37th Annual Computer Software and Applications Conference*, pages 415–424, July 2013. DOI: 10.1109/COMPSAC.2013.69.

X. L. Zhang, C. H. Chi, C. Ding, and R. K. Wong. Non-functional requirement analysis and recommendation for software services. In *2013 IEEE 20th International Conference on Web Services*, pages 555–562, June 2013. DOI: 10.1109/ICWS.2013.80.

Index