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

DONALD E. KNUTH

GERD DOEBEN-HENISCH, LOUWRENCE ERASMUS, ZEYNEP TUNCER

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

AN ACTOR CENTERED APPROACH TO PROBLEM SOLVING IN ENGINEERING

VERSION 19.JULY 2018

UFFMM.ORG

PUBLISHED BY UFFMM.ORG

UFFMM.ORG

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

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

Acknowledgements This book has a long 'conceptual history' leading back to the Philosophy-of-Science studies of Doeben-Henisch 1983 - 1989 in Munich under the guidance of Peter Hinst[1], many intensive discussions between Doeben-Henisch and Erasmus about Systems engineering since 1999, a paper written by Doeben-Henisch and Wagner 2007 [2] with ongoing discussions since then, a lecture by Doeben-Henisch about formal specification and verification in 2010 [3], two papers by Erasmus and Doeben Henisch in 2011 [4], more than 22 regular semesters with the topic Human-Machine Interaction by Doeben-Henisch at the Frankfurt University of Applied Sciences (Frankfurt, Germany)(unpublished) in the timespan 2005 - 2018, two regular semesters with the topic AAI together with Tuncer in SS2016 and WS2016 at the Frankfurt University of Applied Sciences (Frankfurt, Germany) (unpublished), and two workshops with Erasmus in summer 2016 and Spring 2017 (unpublished). Additionally discussions between Doeben-Henisch and Idrissi about AI and AAI since 2015.

[1] He died 10.May 2018.

[2] 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

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

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

1 Introduction

THE TERM 'ACTOR-ACTOR INTERACTION (AAI)' as used in the title of the book is not yet very common. Better known is the term 'HMI' (Human-Machine Interaction) which again points back to the term 'HCI' (Human-Computer Interaction). Looking to the course of events between 1945 and about 2000 one can observe a steady development of the hardware and the software in many directions.[1]

One can observe an explosion of new applications and usages of computer. This caused a continuous challenge of how human persons can interact with this new technology which has been called in the beginning 'Human Computer Interaction (HCI)'. But with the extension of the applications in nearly all areas of daily live from workplace, factory, to education, health, arts and much more the interaction was no longer restricted to the 'traditional' computer but interaction happened with all kinds of devices which internally or in the background used computer hardware and software. Thus a 'normal' room, a 'normal' street, a 'normal' building, a toy, some furniture, cars, and much more turned into a computerized device with sensors and actuators. At the same time the collaborators of human persons altered to 'intelligent' machines, robots, and smart interfaces. Thus to speak of a 'human user' interacting with a 'technical interface' seems no longer to be appropriate. A more appropriate language game is the new talk of 'interacting actors', which can be sets of different groups of actors interacting in some environment to fulfill a task. Actors are then today biological systems (man as well as animals) and non-biological systems. Therefor we decided to talk instead of Human-Machine Interaction (HMI) now of 'Actor-Actor Interaction (AAI)'.

[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.

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

[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

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.

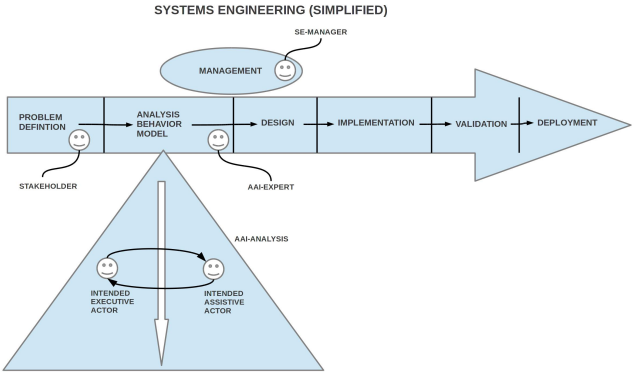

Figure 1.1: Engineering process with diferent kinds of actors

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.

$$AAIA(x) \quad iff \tag{1.1}$$

$$x = \langle A, D, D_P, D_{AAR}, M_{SR}, M_{DIntf}, M^*_{DIntf}, \delta \rangle$$

$$\begin{aligned}
A &:= Set\ of\ actors \\
D &:= Set\ of\ documents \\
D_P &:= Set\ of\ problem\ documents \\
M_{SR} &:= Set\ of\ behavior\ models \\
M_{DIntf} &:= Set\ of\ real\ interfaces \\
M^*_{DIntf} &:= Set\ of\ optimized\ real\ interfaces \\
\delta &= \alpha \otimes \beta \otimes \gamma \otimes o \\
\alpha &: A \times D \longmapsto D_P \\
\beta &: A \times D \times D_P \longmapsto M_{SR} \\
\gamma &: A \times D \times M_{SR} \times D_{AAR} \longmapsto M_{DIntf} \\
\omega &: A \times D \times M_{SR} \times D_{AAR} \times M_{DIntf} \longmapsto M^*_{DIntf}
\end{aligned}$$

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.

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 (FRqs)* of the problem and it can include a set of *non-functional requirements (NFRrs)* distributed throughout the whole actor story.

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. The whole evaluation prrocess 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_e, to talk about his experiences of the empirical world. What Biology and Psychology are telling us is that the *communication* of the experts is grounded in their *cognitive machinery* embedded in their *brains*. Because the human brain in the body is not directly interacting with the outside world but mediated by sensors and actuators the brain constructs an *inner model of the outside world*. And it are exactly the properties of this 'inner model' which provide a 'point of reference' for all our thinking and talking.

One conclusion from these considerations is that the reality for a human person is basically perceived as a *stream of events*, which can be divided

in distinguishable situations, called *states*. A state is understood as a set of properties embedded in a three-dimensional *space*. If at least one property changes a state changes. Subsets of properties can be understood as *objects*, which in turn can be subdivided into '*actors*' and '*non-actors*'. Actors can 'sense' their environment and they can 'respond'. More distinctions are possible as needed.

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

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.

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

Check for AAI-Analysis

The problem given in a systems engineering process must not necessarily be appropriate for an AAI analysis. Therefore it makes sense to do some test in advance whether the problem in a problem document D_P is fitting to an AAI analysis. Such a test of the problem in a problem document checks for the *occurrence* of the following properties:

1. Does the problem include at least one *task (T)* to be realized to reach a solution?
2. Does the problem include an *environment (ENV)* for the task?
3. Does the problem include at least one *executive actor (ExecA)* as the intended user, which shall use some technology as an *assistive actor (AssisA)* – often called *interface* – to run the task?

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

Behavior Model

Following the schema 1.1 further we meet the next sub-process called beta $\beta : A \times D \times D_P \longmapsto M_{SR}$. 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 several 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.
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.
3. The *third* chapter shows how one can test the logical layout of the whole symbolic space given as actor story with actor models by *simulating* the story with the interacting actors as well as by using *automatic verification*, to check specifically specified properties (e.g. some non-functional requirements).
4. In the *fourth* chapter it will be described 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.
5. The *fifth* chapter shows how one can *test* the quality of the *interaction* between the intended user given as a *physical test-person* and the assistive actor given as a *physical interface*. This will be done by a series of usability-tests which enable the generation of a finite set of *tested interfaces*, which allows the identification of a *relatively best candidate*.

3
Actor Story (AS)

To *communicate a story* we assume the following main modes: *textual*, *pictorial*, *mathematical*, as well as *simulation*. Actually it is not clear whether one should prefer the sequence $textual - pictorial - formal - simulation$ or $textual - formal - pictorial - simulation$. Below the first sequence is used.,

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

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

The relationship between the traditional '*functional requirements (FR)*' and the 'actor story' is such, that all necessary functional requirements have to be part of the actor story. The '*non-functional requirements (NFR)*' have to be defined in their intended meaning before the actor story and then it must be shown, how the structure of the actor story 'satisfies' these criteria. In this sense are the 'non-functional requirements' presented as 'constraints' which have the status of 'meta-predicates', which have to be designed in an appropriate 'control logic' for actor stories.[1]

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

[1] This topic of 'Non-Functional Requirements (NFRs)' as well as 'Functional Requirements (FRs)' and their relationship is a hot topic in systems engineering and has not yet a complete solution. The general problem is how to 'represent' the NFRs in a way, that these can be handled in the overall system. The following selected papers (only a subset of thematic related papers) can illustrate the discussion: dealing mainly with NFRs see Khalique et.al. (2017) , Fellir et.al. (2015) , Mairiza et.al. (2013) , Suhr et.al. (2013) , Yin et.al. (2013) , Zhang et.al.(2013) , Menzel et.al. (2010) , Liu et.al. (2012) , Kassab et.al. (2009) . Dealing mainly with FRs see Lian et.al. (2017) , AbrahÃčo et.al (2013) . The big advantage of the AASE paradigm in this context is that the mathematical version of the actor story provides a formal struc-

The *Philosophy behind the Actor-Story concept* – as pointed out in the figure 3.1 – is given in a draft paper by Doeben-Henisch (2018) [2] describing the basic relationships between the empirical external world with the body as a part and the internal, mental, structures and processes enabling things like concepts, memories, languages with meaning etc.

[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/

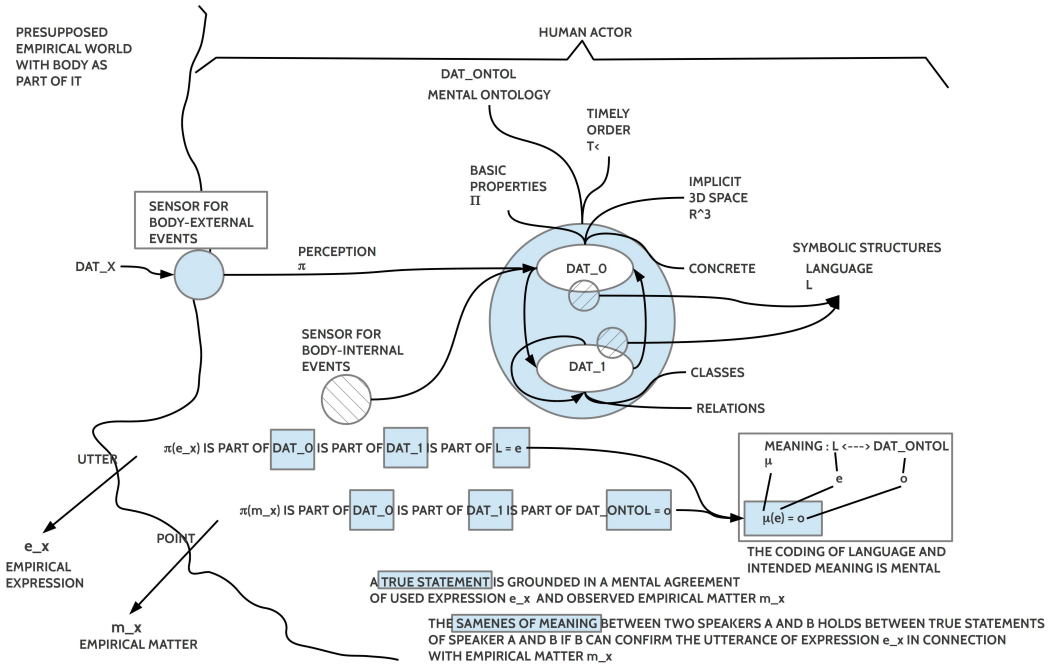

Figure 3.1: Basic Mappings between empirical reality with body as a part and internal mental structures

From this one can derive that different modes to represent empirical matters with symbolic expressions like a language L have as primary point of reference the '*mental ontology*' DAT_{ontol} of the AAI experts. While the mental ontology is assumed to be 'the same' for all different modes of symbolic articulation[3], the different modes of articulation can express different aspects of the same mental ontology more highlighted than in other modes of symbolic articulation.

[3] 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!

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

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

Textual Actor Story (TAS)

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

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

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

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

Every finished actor story has a least one path.[4]

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

Pictorial Actor Story (PAS)

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

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

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

As one can read in the section 1 'Philosophy of the View-Point' (and in the figure 3.1) the world of objects for a *standard user* is mapped into a spatial structure filled with properties, objects, actors and changes. This structure gives a blue-print for the *structure of the possible meaning* in an observer looking to the world with a 3^{rd}-person view. Therefore a pictorial language can *substitute* the intended meaning to some degree if the pictorial language provides real pictures which are structurally sufficient similar to the perceived visual structure of the observer.

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

Mathematical Actor Story (MAS)

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

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

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

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

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

$$\gamma^{+}(g) \quad iff \quad g = \langle V, E, L_{\Pi}, L_{\chi}, \lambda, \epsilon \rangle \tag{3.1}$$
$$E \quad \subseteq \quad V \times V \tag{3.2}$$
$$\lambda \quad : \quad V \longrightarrow 2^{L_{\Pi}} \tag{3.3}$$
$$\epsilon \quad : \quad E \longrightarrow 2^{L_{\Xi}} \tag{3.4}$$

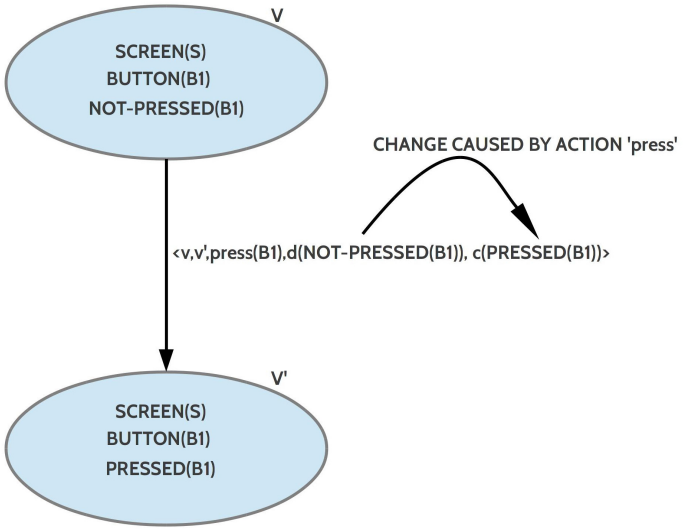

Figure 3.2: Change event between two states

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

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

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

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

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

2. For every 'object' in D^R one can introduce a 'name' realized as a string of 'small alphanumeric letters' beginning with a 'capital letter'. Names are a subset of terms. Examples: *'Hobbes', 'U2', 'Moon', ...*

3. Mappings from distinct objects into other distinct objects which have all to be objects of D^R are called 'functions' realized as a string of

'small alphanumeric letters' followed by n-many terms enclosed in brackets. Functions are as well a subset of terms. Examples: *'add(3,4)', 'push(Button1)', ...'*.

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

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

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

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

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

4. Thus change statements are terms derived as a subset as follows: $\epsilon \subseteq V \times V \times ACT \times \Pi^{Nat} \times \Pi^{Nat}$ (with Nat as the natural numbers including 0).[5]

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

6. Additional to the names of possible objects we assume a special operator *'not(n)'* applied to a name 'n'. The meaning of the operator is, that in this case not the name 'n' is valid, but the 'absence' of the object signified by the name n'. This is important because otherwise in case of many alternative options one has to enumerate all alternatives to an object named 'n'.

[5] The default assumption is that either the delete or the create function has to have at least one property argument.

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

Simulated Actor Story (SAS)

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

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

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

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

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

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

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

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

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*.

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.

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*.

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.

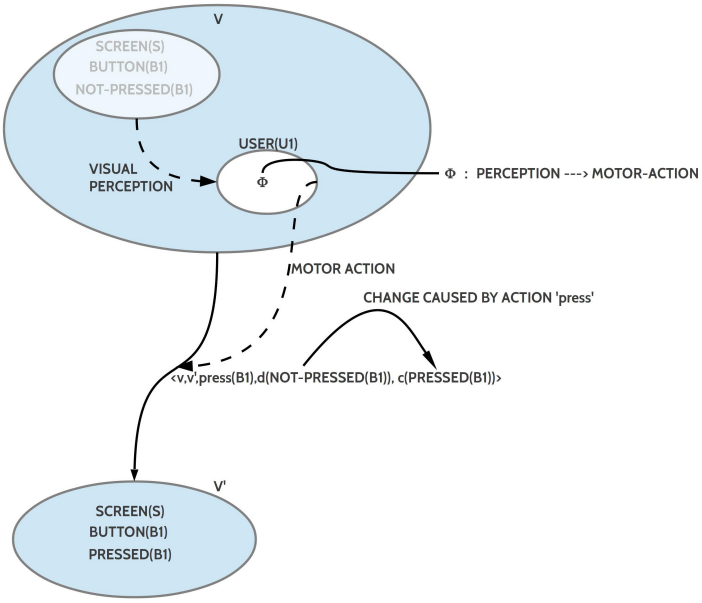

Figure 3.3: Change event with an embedded actor

The general idea of this interaction between actor story and actor model can be seen in figure 3.3.

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'.

4
Actor Model (AM)

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

$$\phi_e \quad = \quad \{(i,o),\cdots,(i,o)\} \tag{4.1}$$

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:

$$\phi \quad : \quad I \longmapsto O \tag{4.2}$$

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*.

$$\phi_e \quad \subseteq \quad \phi \tag{4.3}$$

We understand here a *model* as a mere collection of rules, while an *algebraic structure* is an extension of a model by including additional sets as well as axioms. But we use the term 'model' here equivalently to the term 'algebraic structure'.

Actor as Input-Output System

To enable a *transparent interaction* between actor and environment it will be assumed that an actor is generally an *input-output system (IOSYS)* , that means that an actor has (i) *inputs (I)* from the environment (here the actor story), which are translated by some kind of a 'sensoric system' generating *inputs (I)* for the receiving actor as well as (ii) *outputs (O)* from the actor which can cause changes in the environment. The sum of all inputs I and outputs O defines the *basic interface (BIntf)* of an input-output system S in an environment E.

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

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

It is difficult to describe formally the interaction between an environment (E) and an actor (A). The environment offers existing properties which can change from time to time. The possible 'effect' of these properties and their changes depend on the built-in sensor functions of the actor. Thus the stimulus-function 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 σ_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:

$$\begin{aligned} ENV(E) &:= Environment\ E \\ ENV(E) &\ iff\ E = \langle \Pi, ACT, \sigma, \mu \rangle \\ \Pi &:= Set\ of\ properties \\ ACT &:= Set\ of\ actors \\ ACT &\subseteq 2^{\Pi} \\ \sigma &: 2^{\Pi} \longmapsto ACT(stimulus\ function) \\ \mu &: O_{ACT,resp} \longmapsto 2^{\Pi}(response\ function) \end{aligned} \tag{4.4}$$

and:

$$
\begin{aligned}
ACT(A) &:= Actor\ A \\
ACT(A) &\ iff \quad A \in ACT \wedge A = \langle I, O, IS, \sigma, \mu \rangle \\
I_A &:= Input \\
O_A &:= Output \\
\sigma_A &: rn(\sigma_E) \longmapsto I_A \\
\mu_A &: O_A \longmapsto O_{A,resp}
\end{aligned}
\tag{4.5}
$$

and:

$$
\begin{aligned}
IOSYS(S) &:= Input - Output\ System \\
IOSYS(S) &\ iff \quad S = \langle I, O, IS, \phi \rangle \\
I &:= Input \\
O &:= Output \\
IS &:= Internal\ States(can\ be\ empty) \\
\phi &: I \times 2^{\Pi} \times 2^{\Pi} \times O
\end{aligned}
\tag{4.6}
$$

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'.

Input-Output Systems Basic Typology

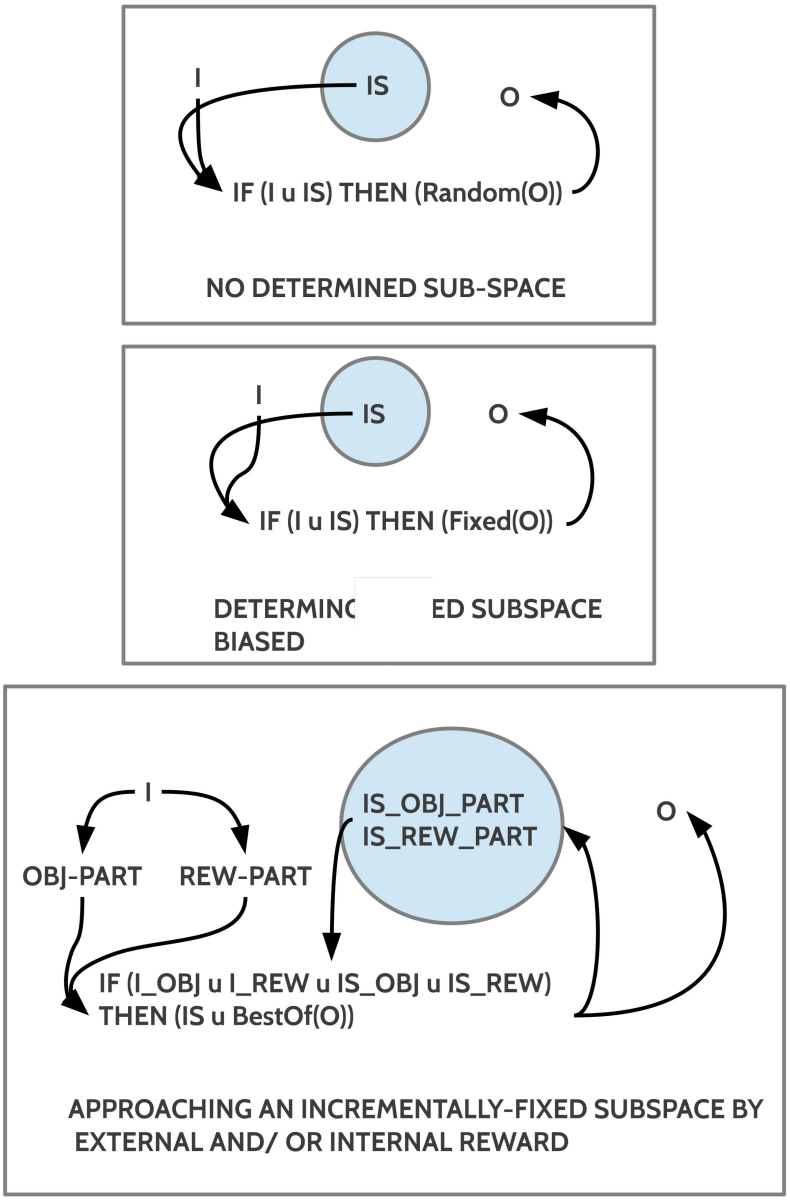

Figure 4.1: Basic Typology of Input-Output Systems

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 4.1).

A first case is the *random* case where the output of a system will be completely random within the space of possible outputs independent of the input. Thus with regard to the set of random possible system-dependent outputs $O_{Random,SYS}$ every output can occur.

A second case is the *fixed (deterministic)* case where a subset of the system-dependent outputs $O_{Fixed,SYS}$ 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.

Empirically and Non-Empirically Motivated

The general definition of a learning input-output offers space for nearly infinite many concrete instances. One possible classification scheme could be that of *empirically motivated* or *non-empirically motivated* models.

Empirically Motivated

Examples of empirically motivated models are some of the models which experimental psychologists have tried to develop. One famous team of psychological motivated researchers was the team Card, Moran and Newell working at the Paolo Alto Research Center (PARC) starting in 1974. They published a book 'The Psychology of Human-Computer Interaction' where they showed how one can develop empirical models of human actors. According to Card et al.(1983)[2] one can assume at least three sub-functions within the general behavior function:

[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

$$\phi = \phi_{perc} \otimes \phi_{cogn} \otimes \phi_{mot} \tag{4.7}$$
$$\phi_{perc} := Perception \tag{4.8}$$
$$\phi_{perc} : I \longmapsto (VB \cup AB) \tag{4.9}$$
$$VB := Visual\ buffer \tag{4.10}$$
$$AB := Auditory\ buffer \tag{4.11}$$
$$\phi_{cogn1} : (VB \cup AB) \times M_{STM} \longrightarrow M_{STM} \tag{4.12}$$
$$\phi_{cogn2} : M_{STM} \times M_{LTM} \longrightarrow M_{STM} \times M_{LTM} \tag{4.13}$$
$$\phi_{cogn1+2} := Cognition \tag{4.14}$$
$$\phi_{mot} : M_{LTM} \longrightarrow O \tag{4.15}$$
$$\phi_{mot} := Motor\ activity \tag{4.16}$$

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:

$$VB \cup AB \cup M_{STM} \cup M_{LTM} \subseteq IS \tag{4.17}$$

The complete model can be found in the cited book.

Non-Empirically Motivated

In many cases non-empirically motivated models are sufficient. This amounts to the task to 'invent' a function 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 4.1) derived from the actor story 'as it is'.

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

GOMS Model

One old and popular strategy for non-empirically motivated models is labeled *GOMS* for *Goals, Methods, Operators* and *Selection rules*[3].

[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

- **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:

$$IF\ I = X \wedge IS = Y \quad THEN \quad IS = Y' \wedge O = Z \tag{4.18}$$

Example: An Electronically Locked Door For the following demonstration we use the simple example of an electronically locked door.[4]

[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

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

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

The details of the different states are given here.

1. Start = U1 ∪ S1 ∪ Env1
   U1 = {USER(U1)}
   S1 = {SYSTEMINTF(S1),KEYPAD(K1), PART-OF(K1, S1), KEY(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 4.6, 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

GOMS Actor Model

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

We will start the construction of a GOMS model for the executive actor using the electronically locked door. For this we simplify the GOMS-Model

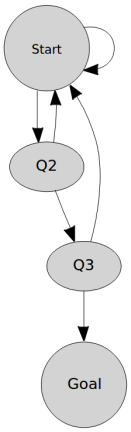

Figure 4.2: Electronic door example - bare graph, only nodes

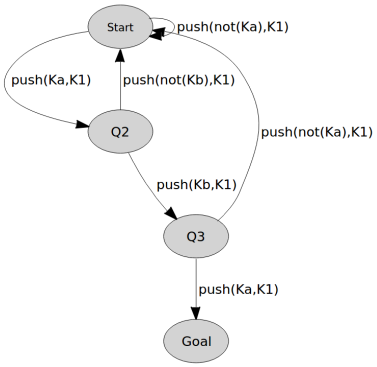

Figure 4.3: aai-example electronic door: nodes and minimally labeled edges

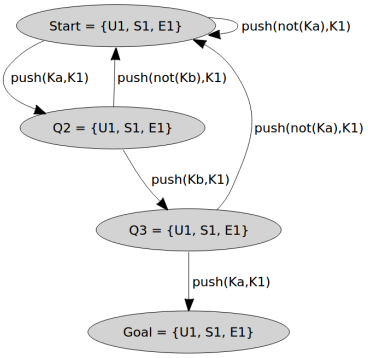

Figure 4.4: aai example electronic door with nodes, shortened edge-labels, and subsets of properties

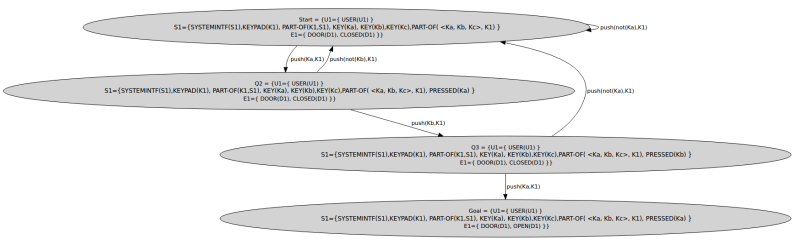

Figure 4.5: aai example with a complete graph (only the edge labels are shortened)

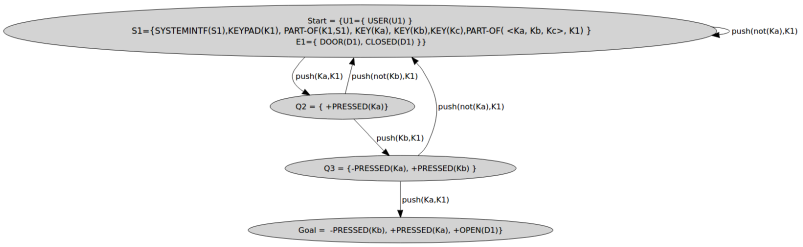

Figure 4.6: Graph with complete start state followed by difference states based on labelled edges

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.

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. *Űber das Gedächtnis: 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*.

5
Simulation and Automatic Verification

This chapter shows how one can test the logical layout of the whole symbolic space given as actor story with actor models by *simulating* the story with the interacting actors as well as by using *automatic verification*, to check specifically specified properties (e.g. some non-functional requirements).

7
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.

8
AS-AM Summary

The Logical Space

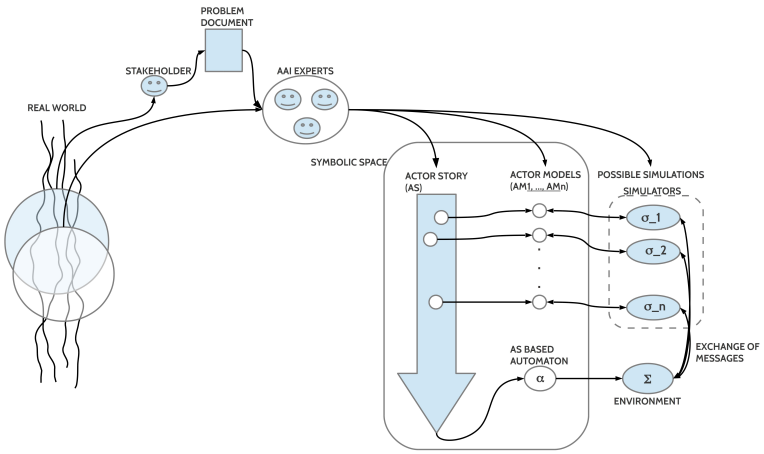

Figure 8.1: The actor story (AS) and the actor models (AMs) as symbolic representations constituting a symbolic space

Looking back to all these new concepts and complex relationships the figure 8.1 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.

Creating the Symbolic Space

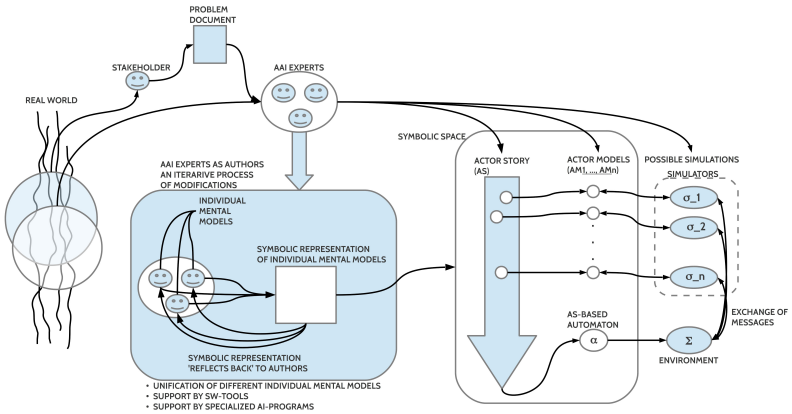

Figure 8.2: Creation of the symbolic space by AAI-experts

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 [1] and figure 3.1) 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.

[1] 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/

The Physical Space

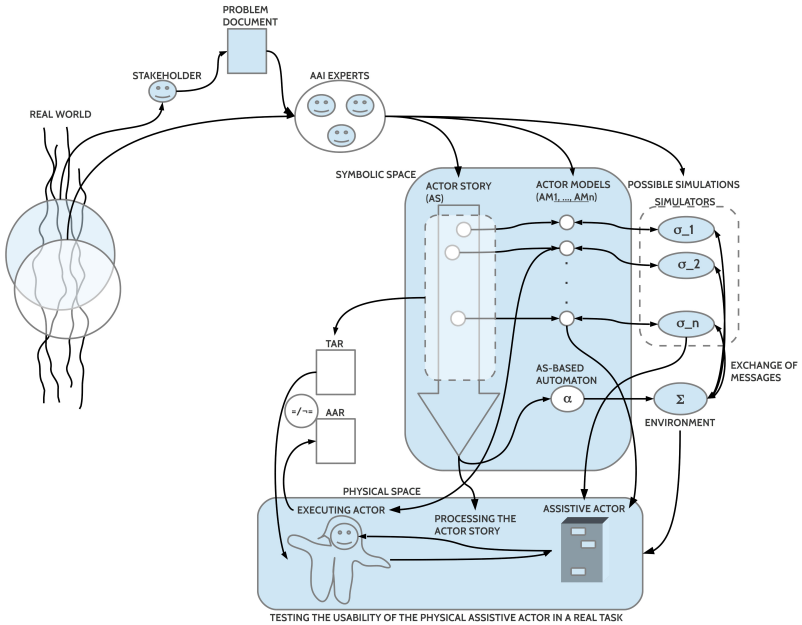

Figure 8.3: The symbolic space extended with simulators translated into the physical space with physical actors as well as physical assistive actors

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 8.3 there are two kinds of actors which have to be grounded in the real world of bodies: the *assistive actor* A_{ass} and the *executing actor* A_{exec}.

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

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

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

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

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

Multiple Actor Stories

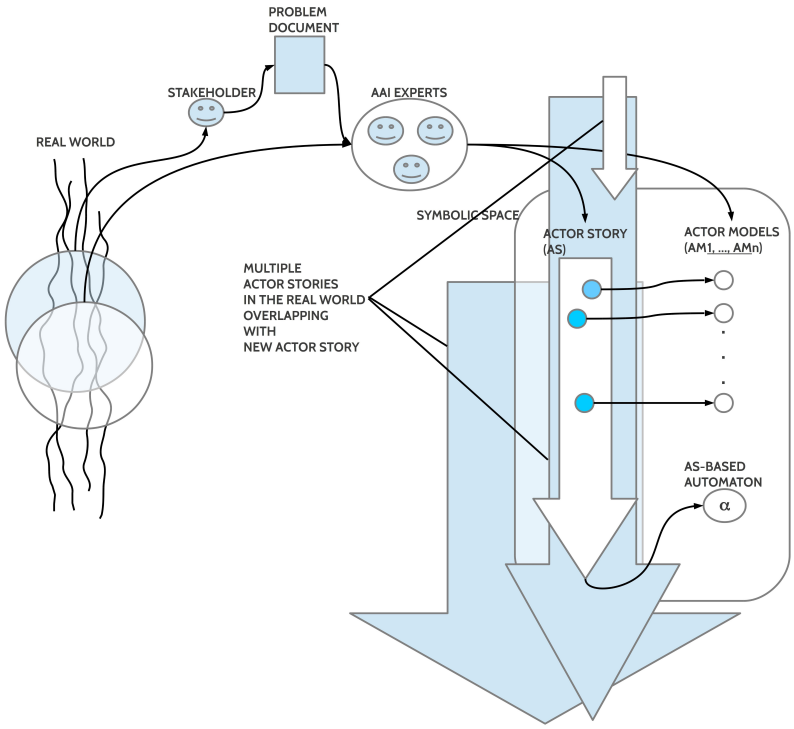

Figure 8.4: The intended actors of an actor story (AS) are living as real actors usually in more than one actor story

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.

9
Looking Forward

Afterthe AAI-Analysis

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

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

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

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

$$v_v \quad : \quad M_{SR} \times M_{SR,real} \longmapsto [0,1] \tag{9.1}$$

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

ToDos

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

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

1. The history part **??** should be refined to make clear to which extend the AAI paradigm has been prepared by different publications, and where have been the differences so far.
2. The different views part **??** should be enriched with more direct comparisons with the main systems engineering positions (especially with INCOSE).
3. The Philosophy of the AAI-Actor part 1 should be extended to give all necessary ideas directly.
4. In the introducing part of the chapter Actor Story 3 one has to explain what non-functional requirements are and it has to be created a new section/ chapter to explain how one can in the AAI-environment give the definition and the check for non-functional requirements in a complete new way.
5. In the subsection of the mathematical AS 3.0.3 one has to rewrite all the used formal languages distinguishing between the mathematical graph as such L_ϵ, the static-property language L_Π attached to the nodes, the dynamic-change language L_Ξ attached to the edges and the specification language of the actor models L_α. The domain of reference D^R has to be clarified for all these different languages.
6. It would be great finally to define also the pictorial language L_{pict} 3.0.2 .
7. Another pending point is the automatic conversion of an AS into a symbolic simulator 3.0.4 which then could be implemented on a real machine.
8. It would be great if we could set up a software framework where we can run the AS-simulator Σ together with all the actor models $\{\sigma_1, ..., \sigma_n\}$; this enhanced with the pictorial language ...
9. It could help to automatize the generation of the TAR document 3.1 based on the AS.
10. There should be some more theory and a manual based on Psychology which describes how to construct an AAR-document. 3.2 3.3
11. It has to be defined more clearly how one can compare AAR documents by distances with the TAR document and how to establish a classification based on these comparisons 7.

12. After rewriting the formal languages for AS and AM the interaction between an AS and the different AMs should be described more precisely 3.4 .

13. In the chapter about AMs 4 there are many points which can be improved:

    (a) More explanations to the concepts 'model' and 'algebraic structure' (better perhaps: 'theory').

    (b) Generally more examples of actor models with different kinds of formalizations, at least one with a common programming language.

    (c) Providing a small algorithm to support the automatic generation of dot-based-graphs 4.3.3 out of the formal description of an AS.

14. Give examples of real usability tests done according to the new protocol 7 .

15. Check the proposed interface between AAI-analysis, logical design, implementation and validation of a systems engineering process 9.1 with the main mechanisms used today in systems engineering.

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

Bibliography

S. Abrah ao, C. Gravino, E. Insfran, G. Scanniello, and G. Tortora. Assessing the effectiveness of sequence diagrams in the comprehension of functional requirements: Results from a family of five experiments. *IEEE Transactions on Software Engineering*, 39(3):327–342, March 2013. ISSN 0098-5589. DOI: 10.1109/TSE.2012.27.

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 Gedächtnis: 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 *ISEM 2011 International Conference*. IEEE, 2011a.

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

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.

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.

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.

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.

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.

X. Lian, J. Cleland-Huang, and L. Zhang. Mining associations between quality concerns and functional requirements. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 292–301, Sept 2017. DOI: 10.1109/RE.2017.68.

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.

I. Menzel, M. Mueller, A. Gross, and J. Doerr. An experimental comparison regarding the completeness of functional requirements specifications. In *2010 18th IEEE International Requirements Engineering Conference*, pages 15–24, Sept 2010. DOI: 10.1109/RE.2010.13.

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.

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