AAI - Actor-Actor Interaction. A Philosophy of Science View

eJournal: uffmm.org, ISSN 2567-6458
https://uffmm.org/2017/10/03/aai-actor-actor-interaction-a-philosophy-of-science-view/

Gerd Doeben-Henisch
info@uffmm.org
gerd@doeben-henisch.de

October 3, 2017

Abstract

On the cover page of this blog you find a first general view on the subject matter of an integrated engineering approach for the future. Here we give a short description of the main idea of the analysis phase of systems engineering how this will be realized within the actor-actor interaction paradigm as described in this text.

1 Introduction

As you can see in figure 1 there are the following main topics within the Actor-Actor Interaction (AAI) paradigm as used in this text[1]:

Triggered by a *problem document* D_p from the *problem phase (P)* of the engineering process the AAI-experts have to analyze, what are the potential requirements following from this document, all the time also communicating with the stakeholder to keep in touch with the hidden intentions of the stakeholder.

The idea is to identify at least one *task (T)* with at least one *goal state (G)* which shall be arrived after running a task.

A task is assumed to represent a *sequence of states* (at least a start state and a goal state) which can have more than one option in every state, not excluding repetitions.

[1]The more traditional formula is known as *Human-Machine Interaction (HMI)*

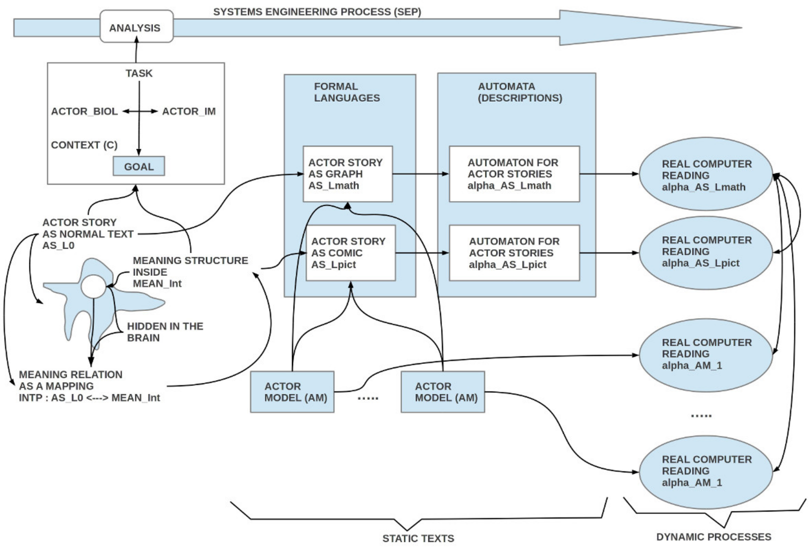
SYSTEMS ENGINEERING PROCESS (SEP)
ANALYSIS
TASK
ACTOR_BIOL
ACTOR_IM
CONTEXT (C)
GOAL
FORMAL LANGUAGES
AUTOMATA (DESCRIPTIONS)
ACTOR STORY AS GRAPH AS_Lmath
AUTOMATON FOR ACTOR STORIES alpha_AS_Lmath
REAL COMPUTER READING alpha_AS_Lmath
ACTOR STORY AS NORMAL TEXT AS_L0
MEANING STRUCTURE INSIDE MEAN_Int
ACTOR STORY AS COMIC AS_Lpict
AUTOMATON FOR ACTOR STORIES alpha_AS_Lpict
REAL COMPUTER READING alpha_AS_Lpict
HIDDEN IN THE BRAIN
REAL COMPUTER READING alpha_AM_1
MEANING RELATION AS A MAPPING INTP : AS_L0 <---> MEAN_Int
ACTOR MODEL (AM)
.....
ACTOR MODEL (AM)
.....
REAL COMPUTER READING alpha_AM_1
STATIC TEXTS
DYNAMIC PROCESSES

Figure 1: Actor-Actor Interaction as static texts and with dynamic computers

Every task presupposes some *context (C)* which gives the environment for the task.

The *number* of tasks and their *length* is in principle not limited, but their can be certain *constraints (CS)* given which have to be fulfilled required by the stakeholder or by some other important rules/ laws. Such constraints will probably limit the number of tasks as well as their length.

1.1 Actor Story

Every task as a sequence of states can be viewed as a *story* which describes a process. A story is a *text (TXT)* which is *static* and hides the implicit meaning in the brains of the participating actors. Only if an actor has some (learned) understanding of the used language then the actor is able to translate the perceptions of the process in an appropriate text and vice versa the text into corresponding perceptions or equivalently 'thoughts' representing the perceptions.

In this text it is assumed that a story is describing only the observable behavior of the participating actors, not their possible *internal states (IS)*. For to describe the internal states (IS) it is further assumed that one describes the internal states in a new text called *actor model (AM)*. The usual story is called an *actor story (AS)*. Thus the actor story (AS) is the *environment* for the actor models (AM).

In this text three main modes of actor stories are distinguished:

1. An actor story written in some everyday language L_0, called AS_{L_0}.

2. A translation of the everyday language L_0 into a mathematical language L_{math} which can represent *graphs*, called $AS_{L_{math}}$.
3. A translation of the hidden meaning which resides in the brains of the AAI-experts into a pictorial language L_{pict} (like a comic strip), called $AS_{L_{pict}}$.

To make the relationship between the graph-version $AS_{L_{math}}$ and the pictorial version $AS_{L_{pict}}$ visible one needs an explicit mapping ι from one version into the other one, like: $\iota : AS_{L_{math}} \longleftrightarrow AS_{L_{pict}}$. This mapping ι works like a lexicon from one language into another one.

From a philosophy of science point of view one has to consider that the different kinds of actor stories have a *meaning* which is rooted in the intended processes assumed to be necessary for the realization of the different tasks. The processes as such are dynamic, but the stories as such are *static*. Thus a *stakeholder (SH)* or an AAI-expert who wants to get some understanding of the intended processes has to rely on his internal brain simulations associated with the meaning of these stories. Because every actor has its own internal simulation which can not be perceived from the other actors there is some probability that the simulations of the different actors can be different. This can cause misunderstandings, errors, and frustrations.[2]

One remedy to minimize such errors is the construction of *automata (AT)* derived from the math mode $AS_{L_{math}}$ of the actor stories. Because the math mode represents a graph one can derive δ from this version directly (and automatically) the description of an automaton which can completely simulate the actor story, thus one can assume $\delta(AS_{L_{math}}) = AT_{AS_{L.math}}$.

But, from the point of view of Philosophy of science this derived automaton $AT_{AS_{L.math}}$ is still only a static text. This text describes the potential behavior of an automaton AT. Taking a *real computer (COMP)* one can feed this real computer with the description of the automaton AT $AT_{AS_{L.math}}$ and make the real computer behave like the described automaton. If we did this then we have a *real simulation (SIM)* of the theoretical behavior of the theoretical automaton AT realized by the real computer COMP. Thus we have $SIM = COMP(AT_{AS_{L.math}})$.[3]

Such a real simulation is *dynamic* and visible for everybody. All participating actors can see the same simulation and if there is some deviation from the intention of the stakeholder then this can become perceivable for everybody immediately.

1.2 Actor Model

As mentioned above the actor story (AS) describes only the observable behavior of some actor, but not possible internal states (IS) which could be responsi-

[2]This problem has been discussed by Doeben-Henisch and Wagner (2007) [DHW07].

[3]These ideas have been discussed in a first version by Erasmus and Doeben-Henisch (2011) [EDH11]

ble for the observable behavior.

If necessary it is possible to define *for every actor* an individual *actor model*; indeed one can define more than one model to explore the possibilities of different internal structures to enable a certain behavior.

The general pattern of actor models follows in this text the concept of *input-output systems (IOSYS)*, which are in principle able to *learn*. What the term 'learning' designates concretely will be explained in later sections. The same holds of the term 'intelligent' and 'intelligence'.

The basic assumptions about input-output systems used here reads a follows:

Def: Input-Output System (IOSYS)

$$\begin{aligned}
& IOSYS(x)\ iff \\
& \quad x = \langle I, O, IS, \phi \rangle & (1) \\
& \quad \phi : I \times IS \longmapsto IS \times O & (2) \\
I &:= Input \\
O &:= Output \\
IS &:= Internal\ states
\end{aligned}$$

As in the case of the actor story (AS) the primary descriptions of actor models (AM) are *static* texts. To make the hidden meanings of these descriptions 'explicit', 'visible' one has again to convert the static texts into descriptions of automata, which can be feed into real computers which in turn then simulate the behavior of these theoretical automata as a real process.

Combining the real simulation of an actor story with the real simulations of all the participating actors described in the actor models can show a dynamic, impressive process which is full visible to all collaborating stakeholders and AAI-experts.

1.3 Testing

Having all actor stories and actor models at hand, ideally implemented as real simulations, one has to *test* the interaction of the elaborated actors with *real actors*, which are intended to work within these explorativ stories and models. This is done by *actor tests* (former: usability tests) where (i) real actors are confronted with *real tasks* and have to perform in the intended way; (ii) real actors are *interviewed* with questionnaires about their subjective feelings during their task completion.

Every such test will yield some new insights how to change the settings a bit to gain eventually some improvements. Repeating these cycles of designing, testing, and modifying can generate a finite set of test-results T where possibly one subset is the 'best' compared to all the others. This can give some security that this design is probably the 'relative best design' with regards to T.

References

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

[EDH11] Louwrence Erasmus and Gerd Doeben-Henisch. A theory of the system engineering process. In *ISEM 2011 International Conference*. IEEE, 2011.