Review of Nancy Leveson
Are you sure your software will not kill anyone?

A Review from the Point of View of the DAAI Paradigm
[*]

Gerd Doeben-Henisch
doeben@fb2.fra-uas.de
Frankfurt University of Applied Sciences
Nibelungenplatz 1
D-60318 Frankfurt am Main

April 2, 2020

Abstract

In an article in the ACM Communications from February 2020 Nancy Leveson describes in a short and concise way the problem of testing systems by their descriptions only.[1] These observations will be taken into account here to check whether and how the DAAI paradigm can deal with these problems.

1 Leveson's View of Save and Reliable Systems

When Leveson speaks about technical systems during their missions she distinguishes clearly between the *physical* and the *software* dimension. Software as *implemented* software has been transformed into physical states inside the physical system, and these physical states *behave* according to an *implicit logic encoded* in a physical set of states *representing* the software.

During mission the *implemented software receives* physical signals from the physical system, is *processing* these signals according to some *implemented logic*, and then eventually *sends* some other physical signals *back* into the physical system. From a logical point of view consists the physical system in mission thus as a synthesis of two physical components: the physical machinery *receiving* signals from the physical *environment* by *sensors*, *generating* events by some *actuators* and processing sensor inputs and actuator outputs by another

[*]Copyright 2019-2020 by eJournal uffmm.org, ISSN 2567-6458, Publication date: April-2, 2020

[1]See Leveson [Lev20]

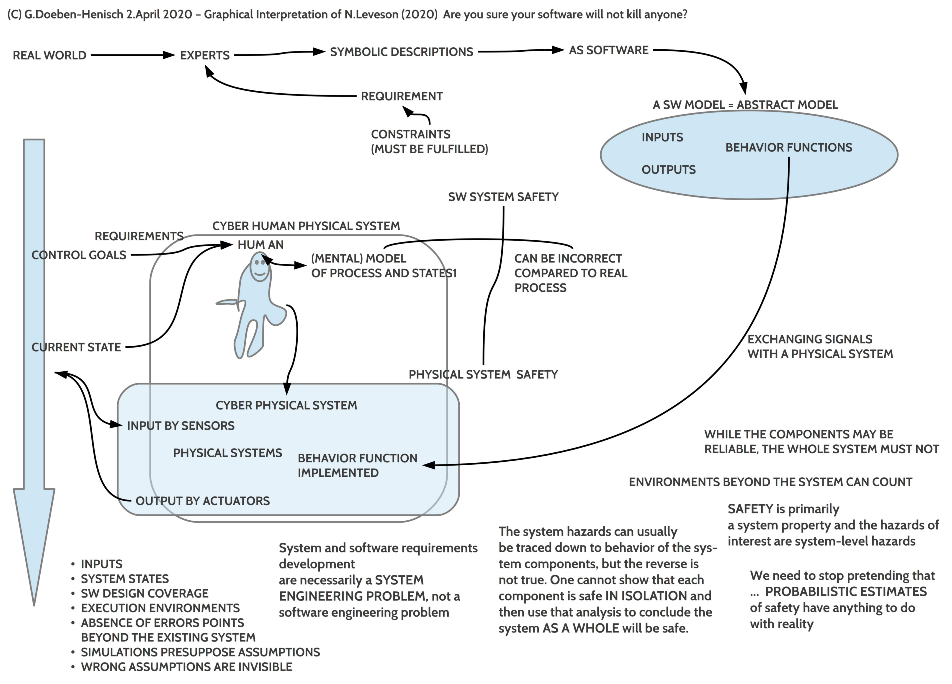
(C) G.Doeben-Henisch 2.April 2020 – Graphical Interpretation of N.Leveson (2020) Are you sure your software will not kill anyone?
REAL WORLD
EXPERTS
SYMBOLIC DESCRIPTIONS
AS SOFTWARE
REQUIREMENT
CONSTRAINTS (MUST BE FULFILLED)
A SW MODEL = ABSTRACT MODEL
INPUTS
BEHAVIOR FUNCTIONS
OUTPUTS
SW SYSTEM SAFETY
CYBER HUMAN PHYSICAL SYSTEM
REQUIREMENTS
CONTROL GOALS
HUM AN
(MENTAL) MODEL OF PROCESS AND STATES1
CAN BE INCORRECT COMPARED TO REAL PROCESS
EXCHANGING SIGNALS WITH A PHYSICAL SYSTEM
CURRENT STATE
PHYSICAL SYSTEM SAFETY
CYBER PHYSICAL SYSTEM
INPUT BY SENSORS
PHYSICAL SYSTEMS
BEHAVIOR FUNCTION IMPLEMENTED
OUTPUT BY ACTUATORS
WHILE THE COMPONENTS MAY BE RELIABLE, THE WHOLE SYSTEM MUST NOT
ENVIRONMENTS BEYOND THE SYSTEM CAN COUNT
SAFETY is primarily a system property and the hazards of interest are system-level hazards
The system hazards can usually be traced down to behavior of the system components, but the reverse is not true. One cannot show that each component is safe IN ISOLATION and then use that analysis to conclude the system AS A WHOLE will be safe.
We need to stop pretending that … PROBABILISTIC ESTIMATES of safety have anything to do with reality
System and software requirements development are necessarily a SYSTEM ENGINEERING PROBLEM, not a software engineering problem
• INPUTS
• SYSTEM STATES
• SW DESIGN COVERAGE
• EXECUTION ENVIRONMENTS
• ABSENCE OF ERRORS POINTS BEYOND THE EXISTING SYSTEM
• SIMULATIONS PRESUPPOSE ASSUMPTIONS
• WRONG ASSUMPTIONS ARE INVISIBLE

Figure 1: Graphical interpretation of Leveson's ideas

embedded physical system which corresponds in its structure to something what computer scientists call a *computer*[2]

In the ordinary case of a *non-learning* system[3] is the implemented logic of the computer-like structure inside the mission system independent of the real-world inputs and outputs. The *observable behavior* of the mission system in the *environment* is connected to the *non-observable* behavior of the embedded software.

On account of this *decoupling* of software induced behavior ϕ_{sw} and environment induced behavior ϕ_{env} it can happen, that the mission system causes failures although the software induced behavior ϕ_{sw} is completely *correct* compared to the *assumptions* giving as *requirements* for the *design* of the software. The *transcendental* logic behind this paradoxical phenomenon of a conflict between the *intention* encoded in software and events caused by the physical behavior of the mission system in the environment reflects a substantial *difference* between the physical environment *as it is* and the physical environment as it is *modeled* in the *cognitive machinery* of the *experts*, the engineers.

This fundamental difference between the real world as it is and the real

[2]Ideally equivalent to the mathematical concept of a Turing machine.
[3]A qualification not explicitly used by Nancy Leveson.

world as transformed into a software system based on *assumptions about the real world* is not only effective during the *construction* of the intended mission system. As Leveson points out it can be effective during the mission too. This happens when a human actor is part of the behavior of the mission system as far as the human actor is part of the control function. The human actor as *executing* actor while *interacting* with the mission system as *assistive* actor is controlling his/ her/ x observable behavior in the environment – with the mission system as part of the environment – by *models* M_{cog} within his/ her/ x *cognitive machinery*. This means that the *perceptions* of the human actor are *interpreted* by his/ her/ x *models* M_{cog}. If these models are in disagreement with the environment as it is then a human actor can cause a human behavior ϕ_{hum} which induces in the mission system signals which will cause a behavior ϕ_{env} of the mission system which causes failures too.

From this follows that the safety or reliability of any distinguishable part of a mission system does not tell anything about the safety or reliability of the whole system. From the point of engineering – including software engineering – the correctness of a system as such can be a complete failure as soon as the whole system will be *deployed* in the environment as it is. According to Leveson this points to a *meta-dimension* beyond the system as such: technical systems today are indeed *technical* systems embedded in a dynamic environment entangled with socio-cultural factors resulting in complex systems of complex systems.

2 The DAAI View of Leveson's View

The interesting question here is whether these interesting observations of Leveson can be handled by the DAAI paradigm.

Following the process as described until now within the DAAI paradigm then one can identify the following phases of the DAAI process, which seem to be relevant for the question:

1. **Problem and Vision:** The DAAI process starts with the generation of a *problem* and a *vision* statement. The usual authors for these statements are the *stakeholders* (which can be everybody). Therefore the DAAI process depends in its root from the *world view* of certain persons which by some *reasons* – conscious as well as unconscious ones – qualify a certain situation as *not optimal* characterizing a problem and as something *desirable* characterizing a vision. The DAAI process as such does not *judge* about the *reasonableness* of these qualifications. Indeed the assumptions encoded in the problem and vision statements can be completely nonsense; the DAAI process as such will not investigate this.

2. **Analysis:** During the DAAI analysis following the problem and vision statement the acting *DAAI experts* will try to analyze all the conditions which have to be fulfilled to enable task driven processes which allow certain actors to do some jobs interacting with other actors in some assumed environment. Inevitably these experts are acting according the *world views* – some set of cognitive models – in their brains. Clearly they can change these world views during their work, but finally whatever will be the outcome of their analysis, the outcome will correspond to the models which are *active* in their cognitive machinery for *interpreting* their perceptions and for *deducing* the details of their analysis. Therefore, if these expert models in their cognitive machinery are in some sense *inadequate* compared to the real world as it is, then the results of the analysis can be inadequate too.

3. **Testing:** There are many different kinds of tests possible. Basically every *test* is a measurement operation comparing something *new* according to some *pre-defined norm*. If the new does *sufficiently well match* with the pre-defined norm one usually interprets this as *fulfilling the test*.

4. **Usability test:** An often used kind of test is called *usability test*: one wants to check whether a defined system as part of a defined process is *good enough* that a *human user* can work in this process in a *satisfying* way with regards to a multitude of criteria. In the DAAI paradigm a distinction is made between the *task induced actor requirements [TAR]* and the *actor actor induced requirements [AAR]*. The TAR can be inferred from the task description of the analysis and results in a behavior profile of an *ideal* actor, an actor which is *wanted* according to the task description. It is completely unclear whether such an *ideal actor* really exists. Calling for potential *real actors* can eventually lead to a set of *candidates* where every individual candidate has its *individual real profile* of possible behavior which represents the AAR. Usually there exists a difference between the TAR and the AAR. For a company the simpler case would be if the AARs of the candidates would *sufficiently match* the TAR of the intended process. If not, it can be an interesting question whether a candidate is able to *change* its actual AAR_t to a new $AAR_{t'}$ which is sufficiently close to the TAR. This presupposes that a candidate is *able to learn* by *experience*.

5. **Usability: not free or free:** If the candidate should be an employee then in some sense the owner of the production process can to a certain degree *dictate* the severity of the TAR requiring that every candidate has to match the TAR sufficiently well otherwise the candidate will not be accepted. This is the case of a *not free* testing process. If the process which

shall be tested is an *optional process* as a product or service which has to be sold to possible customers which *can* by the product/ service but must not, than the testing is *free*: the results of the test can be used to *criticize* the actual process leading to some possible improvements. Thus if the assumptions of the DAAI experts leading to a certain format of the task processes are somehow *deviant* from the reality of possible customer behaviors then a *free* usability test can help to detect wrong assumptions and can therefore principally enable some improvements. If the usability testing is *not free* then companies can produce bad production processes compared to the abilities of real people without being challenged to improve these.

6. **Testing More:** While the usual usability test offers some chance to improve a task process[4] it is completely unclear how other possible wrong assumptions which are already part of the problem and vision statements can be detected and in some sense being *improved*. Although Nancy Leveson does not characterize exactly where in a possible engineering process wrong assumptions can be introduced, but one can easily imagine that the problem and vision statements represent a vast space of possible wrong assumptions. And, not to forget, the experts themselves, are not free from wrong world views whose *wrongness* as such is not visible. *Not knowing* something or *knowing* it in the *wrong way* is usually not easily detectable. This points to a fundamental problem of using knowledge in a society: the wide spread acceptance of *being correct* appears in this systems engineering context as *very dangerous*! Being correct is only partially a good strategy to cope with an unknown future.

References

[Lev20] N.G. Leveson. Are you sure your software will not kill anyone? *Communications of the ACM*, 63:25 – 28, 2020. https://doi.org/10.1145/3376127.

[4]But not necessarily if the criteria used during the test are inadequate!