A Requirements-based Framework for the Analysis of Socio-technical System Behaviour

Jon G. Hall
Open University, Milton Keynes (UK)

Andrés Silva
Universidad Politécnica de Madrid (Spain)

Abstract

Requirements Engineering's theoretical and practical developments typically look forward to the future (i.e. a system to be built). Under certain conditions, however, they can also be used for the analysis of problems related to actual systems in operation. Building on the Jackson/Zave reference model [2] for requirements and specifications, this paper presents a framework useful for the prevention, analysis and communication of designer and operator errors and, importantly, their subtle interactions, so typical in complex socio-technical systems.

1. Introduction

There are three system 'dimensions' that come into play in the analysis of socio-technical systems: these are system structure, system behaviour and human-computer interaction. Many techniques for the analysis and prevention of accidents in socio-technical systems focus on only one or two of these dimensions but not on all three at the same time. However, it is well known that accidents combine all three [1, 5].

In this paper we propose an analytical framework for socio-technical systems. The framework is based upon a dynamic view of the reference model (RM) for requirements and specifications [2]. The formal approach of the RM, complemented with the semi-formal approach of Jackson's Problem Frames [3], provides a clear distinction between descriptions of the world (or environment of the system) W, the requirements R and the specifications S of a solution machine. This specification S contains descriptions expressed exclusively in terms of shared phenomena between the machine and its environment. For a machine to satisfy the requirements, $W \wedge S \models R$ must hold.

With the purpose of including the analysis of system-operator interactions, our framework also contains elements that resemble those of Norman's *Mental Models* [6]. However, Norman concentrates on the design of the system in non-formal terms; our location in mission- (including safety-)critical systems requires reasoning capability above the *ad hoc*. As with Norman, we work with three different viewpoints: two mental (corresponding to design and operating stakeholders), one physical (the system in operation). Those stakeholders can hold discrepant views of W, S, R [8] that, through their interactions, can lead to undesirable physical situations. Our intention is to place stakeholders within the same spatio-temporal framework to allow proper analysis of them and the physical system individually and severally. Although we admit design and operating *teams*, throughout this paper we use the singular, for consistency.

2. Designer and operator viewpoints

Our framework is summarised in Figure 1 which should, initially, be seen as three concentric squares. We explain, first, the middle square, corresponding to the actual physical execution of the system. According to the reference model, the system "moves" the environment from a state described by W at time t (abbreviated to W^t in the figure) to a state[1] at $t+1$ (W^{t+1}, which, ideally, satisfies requirements R), as indicated by the mapping $W^t \rightarrow W^{t+1}$. This step is decomposed into three: *(i)* a "sensor"

[1] We assign unit time to the change for simplicity only, it could, in fact, be any delta.

step: the state of the world is input to the software ($W^t \rightarrow S^t$), *(ii)* a software step: the software produces output from its input ($S^t \rightarrow S^{t+1}$) and *(iii)* the "actuator" step: the software output effects some change in the world ($S^{t+1} \rightarrow W^{t+1}$). The other squares correspond to the views of our identified stakeholder: innermost the operator's view (subscript o); outermost the designer's view (subscript d). In this way, we enable the systematic forwards or backwards analysis of any discrepancies between the operator's and designer's views of a step (or sequence of steps) and/or the actual step(s) that took place.

A *delta expression* (shown in the in the figure as a decorated Δ, i.e. Δ_o^{SW}) represents the gap between the operator's view of a step and the step itself. So, for instance, Δ_o^{SW} represents the operator's view of an "actuator" step against reality. Discrepancies in a step can either be in the step itself, or derived from discrepancies between actual states and designed or perceived ones. This second class of discrepancies we label with decorated Xs.

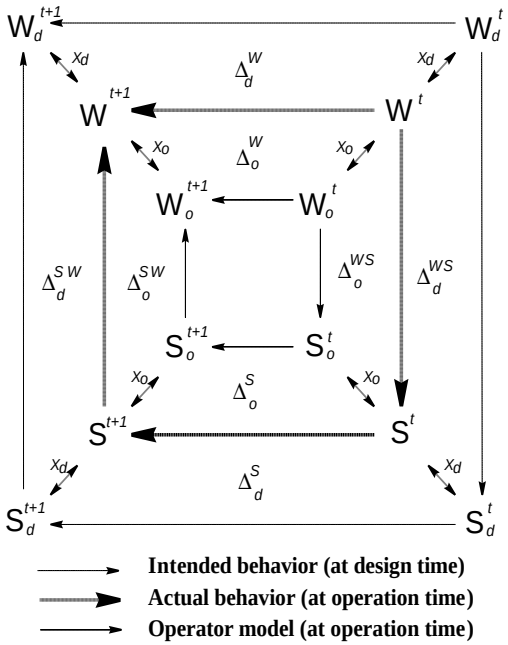

Figure 1. Resume of the framework

One of the immediate advantages that our framework provides is its taxonomic capability. For example, from the delta expressions, *operator errors* can be classified into one of:

- Δ_o^W the operator misconstrues an environmental step;
- Δ_o^{SW} the operator misconstrues an actuator step;
- Δ_o^S the operator misconstrues the sensor/actuator linkage;
- Δ_o^{WS} the operator misconstrues an sensor step.

This is a more granular and detailed view of Norman's mental model (mis)constructions [6]. Also, similar taxonomies exist for misconstrued states (the X's) and for design errors (the Δ_d's), not shown here for reasons of space.

3. Case study: the chemical reactor

This section shows, through a real-world example, how our framework may be used in the analysis of the chemical reactor explosion [4]. The requirements for a chemical reactor, a schematic for which appears in Figure 2, included the clause that when a component in the plant reported a problem, the software should send a warning message to the operators and stop executing. On one bad day, the operator sent an order to the software to open the catalyst (with the aim of increasing the output of the reactor). The program was instructed to, first, open the catalyst and, second, open the flow of cooling water, to regulate the reaction. An accident occurred when a component gearbox reported low oil levels to the software, just after opening the reactor, causing the software to stop. As a consequence, opening the cooling water was never performed. The temperature of the reactor increased resulting in an explosion.

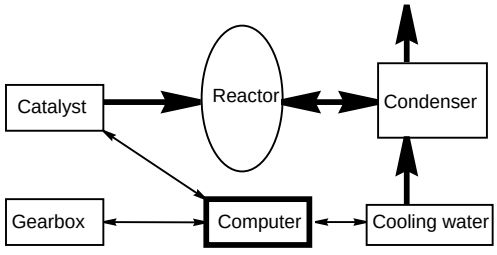

Figure 2. Chemical reactor schematic

For our analysis, we will use a formal approach, related to individual traces of behaviours in, respectively, the real world (i.e., the actual behaviour), the

designer's view and the operator's view. However, less formal approaches could also be used under our framework (for example, based on problem frame concerns[3]).

In the simplest of safety and liveness terms, the situation $\{catalyser\_open, \neg water\_open\}$ should never occur. Figure 3 represents, step by step, the sequence of events. Analysis goes backwards in time, from top to bottom. The three columns represent the designer's view, the actual events and the operator's view of those events. States are shown in angle brackets, where shared phenomena are under the line.

There is a X_d difference, shown at the top of the figure, between the (actual) hazardous state and the state envisioned by the designer. If we trace back in time the sequence of events we find that, previously, in the S^{t+1} state, there are some differences but there are no differences in the previous one (S^t). So it is in this $S^t \rightarrow S^{t+1}$ step where a Δ_d^S actually happened (i.e., the software did not achieve its intended action, even when sensors, actuators and the operator behave correctly). Analyzing this step, what we find here is a design decision that proved to be wrong: the atomicity of the operations of opening the catalyser and the water when, actually, the water was never opened, as the second column shows.

Please note that our framework would admit an alternative analysis if the designers view ignored the possibility of *unit_needs_service* in W^t. In this case, a difference would appear in the W^t stage, indicating that an expectation X_d of the designer was wrong, but at the end, the conclusion would be the same: the program has been poorly designed.

The actual behaviour of the world, machine and operator is expressed by the relation $\models$ on the W, S and R rules, where

R	`if` $(catalyser\_open)$ `then` $water\_open$
	`if` $(unit\_needs\_service)$ `then` $(warn\_operator; STOP)$
	`if` $(req\_catalyser\_open)$ `then` $catalyser\_open$
S	`if` $(openc_{sen})$ `then` $(openc_{act}; openw_{act})$
	`if` $(req\_service_{sen})$ `then` $(warn\_operator_{act}; STOP)$
W	`if` $(openc_{act})$ `then` $catalyser\_open$
	`if` $(warn\_operator_{act})$ `then` $warn\_operator$
	`if` $(openw_{act})$ `then` $water\_open$
	`if` $(unit\_needs\_service)$ `then` $req\_service_{sen}$
	`if` $(request\_catalyser\_open)$ `then` $openc_{sen}$

This model satisfies the RM in the sense that $W \wedge S \models R$ holds[2]. On the other hand, the designed behaviour (W_d, R_d and S_d) differs from this model. Although $W_d = W$ and $R_d = R$, S changes to:

S_d	`if` $(openc_{sen})$ `then` $[openc_{act}; openw_{act}]$
	`if` $(req\_service_{sen})$ `then` $(warn\_operator_{act}; STOP)$

where $[a; b]$ indicates that a followed by b should be atomic, i.e., that no event should come between their occurrence. Similarly, rules and relation $\models_o$ could be provided for the operator; however, due to the informality of the operator (it is a *biddable domain* in Jackson's terms[3]) we prefer to analyse it informally as follows:

Referring back to Figure 3, from the point of view of the operator the first departure from actual behaviour happens in W^t: the operator ignores that, at the same time he is requesting the catalyzer to open, the gearbox signals a request for service. The operator continues, under the assumption that there is no abnormal behaviour, and is therefore never aware of the hazard, as can be seen in the W^{t+1} row. Although it is not "the solution", this suggests that a proper feedback mechanism for the gearbox request for service is missing.

Although we have been able to present only a simple analysis, we claim that our framework has the potential to raise awareness about many real design and operator errors as well as suggesting solutions by which they can be avoided. To support this claim, consider for instance, the long sequence of $\neg water\_open$ facts without the operator being aware of it, even after his request was made. The analyst could ask why this happens, if it has some importance or not, and how can be avoided in future designs. Also, an analysis of mixed causes (intermingled design *and* operator errors) can be done, like when even subtle design errors lead to wrong information in a display that lead the operator into taking a bad decision [5].

4. Conclusions and Further Work

We have presented a framework into which many aspects of socio-technical systems fit, and within

[2]The reader can check this by chasing round Figure 1 the sequence of events.

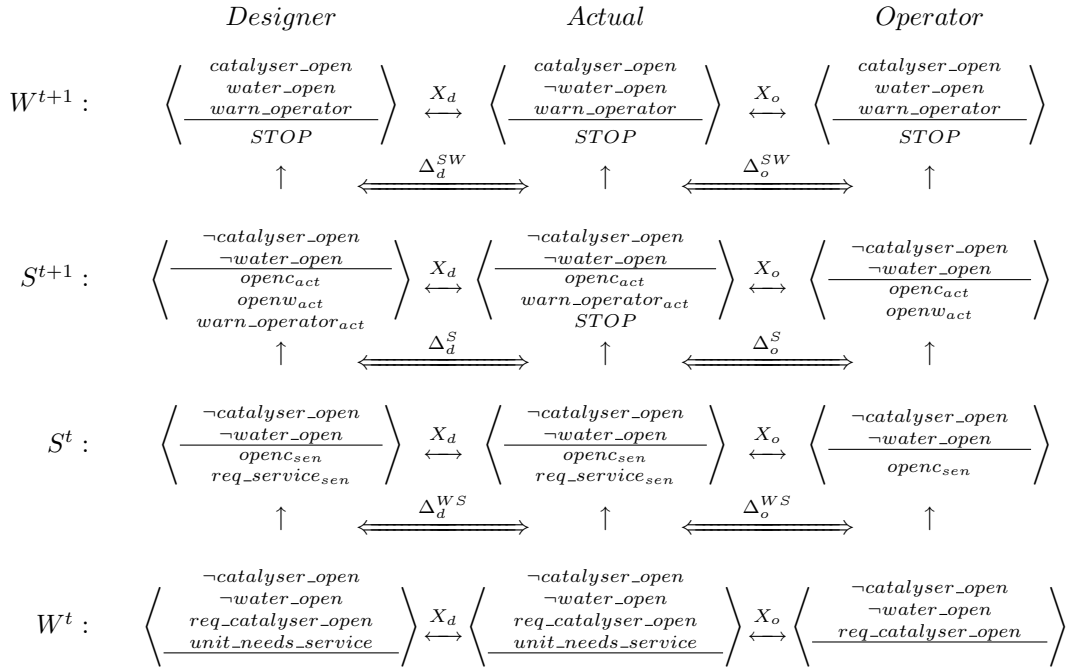

Figure 3. Analysis of the chemical reactor accident

which their interrelationships become clear. With its strong foundations in the reference model, from a safety engineering point of view, we aim in further papers to show how widely used techniques (fault tree analysis (FTA), HAZOP and Event Tress (ET), for instance) can be embedded within our framework. For instance, our Framework provides the means to structure FTA analysis as, in FTA, the backward chaining sequence from the root of the tree (the hazard) can be guided by stepping backwards in the cycle shown in Figure 1. For each step, the operator and designer view can be interleaved to study their interactions, improving former proposals like [1]. In HAZOP, on the other hand, our framework provides a general structure for HAZOP meetings [7]. In this way, HAZOP keywords can be applied to each small step in the cycle, both from a designer point of view and/or from an operator point of view. A similar argument can be provided with regard to ET or other techniques. In this way, our framework provides a coherent view that underlies hazard analysis techniques, also integrating the interaction between operator and design errors.

This framework provides the basis of our future work, whose mission is to bring a complete and coherent view of the misbehaviours in socio-technical systems. Another approach we are exploring is the potential use of our framework for storing and distributing accident-related information.

References

[1] C.F. Fan and Chen W.H. Accident sequence analysis of human-computer interface design. *Reliability Engineering and System Safety*, (67):29–40, 2000.

[2] C. A. Gunter, E. L. Gunter, M. Jackson, and P. Zave. A reference model for requirements and specifications. *IEEE Software*, 17(3):37–43, 2000.

[3] M. Jackson. *Problem Frames: Analyzing and structuring software development problems.* Addison-Wesley, 2001.

[4] T. Kletz. *Computer Control and Human Error.* Institution of Chemical Engineers, 1995.

[5] N. Leveson, L. D. Pinnel, Sandys S.D., S. Koga, and J.D. Reese. Analyzing software specifications for mode confusion potential. In *1st ACM SIGSOFT Symp. on the Foundations of Software Engineering*, Dec. 1993.

[6] D.A. Norman Cognitive Engineering. In D.A. Norman, S.W. Draper, *User Centred System Design.* LEA Associates, New Jersey, 1986.

[7] F. Redmill, M. Chudleigh, and J. Catmur. *System Safety: HAZOP and Software HAZOP.* Wiley, 1999.

[8] A. Silva. Requirements, domain and specifications: A viewpoint-based approach to requirements engineering. In *International Conference on Software Engineering 2002 (ICSE 2002)*, 2002.