Available online at www.sciencedirect.com
ScienceDirect
Safety Science xxx (2006) xxx–xxx
SAFETY SCIENCE
www.elsevier.com/locate/ssci

A conceptual model for the analysis of mishaps in human-operated safety-critical systems

Jon G. Hall [a], Andrés Silva [b,*]

[a] *Centre for Research in Computing, The Open University, Walton Hall, Milton Keynes MK7 6AA, United Kingdom*
[b] *Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, Boadilla del Monte 28660, Spain*

Received 12 June 2006; received in revised form 24 October 2006; accepted 4 November 2006

Abstract

In this paper we provide a conceptual model useful for describing the cyclic interactions of a computer system with its environment and with the humans operating it. Furthermore, it describes how undesirable events introduced at operation time and/or at design time can interact, possibly leading to catastrophic consequences. The model provides a basis for the proper design and analysis of safety-critical systems with human and computer-based components. The model is derived from the requirements engineering reference model of Zave and Jackson by the addition of behavioural dynamics, the inclusion of an operator and feedback. The model looks far beyond individual failures and considers problems not as the chaining of events but as the chaining of deviations from design, from operation and from their interaction. Our goal is a model through which analyses that transcend the traditional reductionist approach in event-chain models can be conducted.

Keywords: Software safety; HCI; Control; Accident models; Hazard analysis; Risk analysis; Human factors

1. Introduction

In 1979, at Three-Mile-Island (TMI), the operators were confused by a warning light that wrongly indicated a valve to be open when it was not. Their confusion was a

* Corresponding author. Tel.: +34 913366921; fax: +34 913366917.
E-mail addresses: J.G.Hall@open.ac.uk (J.G. Hall), asilva@fi.upm.es (A. Silva).

0925-7535/$ - see front matter
doi:10.1016/j.ssci.2006.11.001

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

contributory factor in the TMI accident. In 1988, an Aegis operator had intended to target an Iranian F-14 fighter but targeted a civilian Airbus instead. The Aegis interface did not properly identify the targeted aircraft, confusingly displayed its behaviour (the F-14 was actually on a runway) and provided no clues that would help to identify such a mistake. The Airbus was brought down with 290 people aboard. In December 2001, several soldiers were killed by a bomb targeted on their position because, after replacing a dead battery in a device to determine the target, the device reset the GPS position to its own. In December 2004, an F/A-22 crashed on takeoff in Nevada. The pilot lost control of the plane because of a failure in the digital flight control system. The pilot apparently received no warning that a failure had occurred. (Detailed accounts of these and other accidents can be found in Neumann (1994) or in the Risks Digest web-site, www.risks.org.)

In accidents of this type, operator error is often quickly (and conveniently) identified as the cause. Of course, the reality is somewhat more complex. In these examples, and many other cases, operator actions were confused and compounded by component and/or design errors in such a way that, basically, the operator did not stand a chance.

Three classes of event played a role in these accidents:

- events that happen in the outside world;
- events that happen within the safety-critical technical system; and
- operator-related events.

For each of the examples above it is not events from any one class that were the cause of the accident but, importantly, it is the interaction of events from each of the classes. For instance, in the dead battery accident, events in the outside world – the fact that the bomber crew asked for a second calculation in degree decimals, forcing the soldiers to use the now dead-battery GPS receiver a second time – were combined with system events – the dead battery – and operator-related events – trusting the targeting device after replacing the battery. A view of accidents that diminishes the importance of any of those three classes of events or of their interaction oversimplifies.

In any system that aims to control certain aspects of its environment there will be a cycle of interactions of the system with its environment and vice-versa. This we will call the *system–world* cycle. When an operator is involved we must also account for their influence on system dynamics, rather than seeing them as just another component in the environment of the system. This leads us to propose the *system–world–operator* cycle which intertwines the system–world cycle with the *system–operator* cycle. The system–world–operator cycle brings with it considerable challenges for the designer of a system, specifically in taking into account the possible interactions between its constituent cycles.

1.1. Interaction challenges

Confusion and mistakes can happen at design time as well as at operation time. Here we make the observation that, whereas conceptual problems at design time may influence the system in operation, it is not the case the other way round. As in the examples above, poor system and feedback design can lead to dangerous chains of confusion during operation, even ending in disaster. The effects of poor design at operation time cascade as erroneous designer and/or operator views of the system diverge from reality or as the operator's actions diverge from what was expected and intended. It is often hard to distinguish oper-

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

ator error from designer error precisely because their interactions are subtle and complex during operation. This raises three challenges for those involved in safety-critical systems:

The challenge for the designer: At design time, the designer of a safety-critical computer-based system needs to focus not only on the system (machine) behaviour but also on its interaction with the environment and with the operator: the challenge is to design a system that (a) performs its intended function, and (b) avoids, prevents or blocks undesirable interactions between the system and its environment and operator.

The challenge for the operator: At operation time, the operator of a safety-critical computer-based system receives feedback from the system and the environment, providing input to the system: the challenge is to interpret the information displayed by the system to create an accurate (or at least benign) view of its actual state, and to interact with the system in order to achieve the specified goal, all without compromising safety.

The challenge for the safety analyst: During safety analysis (or assessment) of a computer-based safety-critical system the challenge for the analyst is to retain an adequate focus *simultaneously* on the operator, the system, and the environment. Furthermore, to be able to analyse a particular event sequence and the cause–consequence relationships between events, the analyst should consider the interactions between system, operator and environment events. In particular, it is not enough to focus on the "human path" or the "system path" of events, as those paths do not exist in isolation.

1.2. Importance of the problem

This challenge of dealing with complex and compounded errors has been repeatedly pointed out in the literature. As early as 1962, Edwards said that decisions taken by an operator are embedded in a wider context (Edwards, 1962): the operator interprets the state of the system in terms of possible actions, chooses one of those actions and the result of executing it is the background for the next action. Errors are difficult to locate in this stream of actions, especially if the operator is exploring the effects of an action in order to achieve an improved system behaviour. James Reason, in his 1990 study based on many examples, illustrated the fact that accidents are often caused by human error *induced* by poor design (Reason, 1987). Taylor, in 1994, suggested that an integrated approach is needed for considering the interaction between software, hardware and the operator (Taylor, 1994). His proposal, however, was based on the juxtaposition of old techniques instead of the analysis of the emerging needs, complexities and subtleties of human–system–environment interaction. The need for a representation of both human error and system failure was still apparent, as was pointed out by Johnson (1998). This observation, although welcome, begged the question of which should be the conceptual framework in which this representation will work. Johnson also called for the integration of Software Engineering techniques with those of Human Factors and Systems Engineering, as well as the consideration of organisational factors (Johnson, 2002), arguing that the advantages obtained would benefit not only accident analysis but also the distribution and understanding of the information gathered from those analyses.

A survey by Johnson on Causal Analysis Techniques (Johnson, 2003) concluded just how ad hoc the mix of human and system failure is as, in any flow of events, events are considered and traced backwards or forwards, informally, and without any guidance on how to do it. Cacciabue (2000) explained how important is to have models of human–machine interaction that are able to couple with representations of machine and working

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

context. Leveson (2004), in her critique of the inadequacy of traditional event-chain models for accident analysis, pointed out the influence of design errors or omissions on some operation-related flaws (such as inadequate feedback), and the consequentially inadequate enforcement of safety. A recent book by Cacciabue that focuses on human factors in complex systems also points out similar problems (Cacciabue, 2004).

The goal of this paper is to provide a reference model through which it is possible to reason about designer–system–environment–operator interactions, especially those that are undesirable. Derived from this reference model is our conceptual framework for the characterisation of cause and effect that can underpin both backwards and forwards analyses of operated safety-critical systems. Our conceptual framework emphasises the role of the designed interface in such systems and the possible consequences that they can have on safety.

1.3. Resumé of the proposal

To meet the challenges set above, we propose a reference model for the analysis of human–system interaction and of system–environment interaction that is able to distinguish between problems introduced at operation time and those introduced at design time. Our proposal is:

(1) to take a tried-and-tested reference model for requirements and specifications (based on (Gunter et al., 2000)) (see beginning of Section 2);
(2) to express it in dynamic terms (Section 2.1);
(3) to add the operator and designer's viewpoints (Section 2.2);
(4) to analyse any undesirable interaction between design time and operation time (Section 3); and
(5) finally, to define representations that can be of practical use in different situations, such as hazard analysis, safety assessment, accident investigation and even training (Sections 4 and 5).

We do not claim that our conceptual framework provides an approach to full accident investigation,[1] at least in the sense that it will not help (nor impede!) the analysis of the organisational and/or social roots of an accident. We are content, for now, to provide a model for human–system–environment interaction that helps in the identification and classification of the events involved in safety-critical situations and that begins (i) to overcome the problems with other event-based approaches, as discussed above; (ii) to take into account problems related to the design of the human interaction with the system, and (iii) to provide help for both retrospective and prospective analysis.

1.4. Paper structure

The paper is structured as follows. Section 2 presents the basis of our framework; Section 3 describes how mishaps can be thought of as discrepancies between the designer, the operator and the real world; Section 4 provides an overview of the benefits of the frame-

[1] Also known as macroscopic-type root cause analysis, according to Cacciabue (2004).

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

work with particular emphasis on how it can be used to overcome our three challenges: that of the designer, that of the operator and that of the safety analyst; Section 5 describes the potential applications of the framework; Section 6 relates our work to the existing literature; finally, Section 7 derives some conclusions.

2. Conceptual basis: three views of the *WSR* model

The basis of the work reported here is found in Zave and Jackson (1997) in which is characterised the notion of completeness of (software) requirements engineering. There are four elements of the *WSR* model. The first three sets of descriptions:

- W: descriptions of the behaviour of the environment of the system–the world into which the system will be deployed. The descriptions are *indicative*, i.e., indicating fact, and thus are true of the world independent of the system.
- S: descriptions of the behaviour across the interface of the solution system with the world.
- R: descriptions of the requirements for how the system in context should behave. The descriptions are *optative*, i.e., they express how the world should behave after the system is deployed.

The fourth element[2] is a proof that S in its context W will satisfy the requirements R; symbolically:

$$W, S \vdash R$$

What is omitted from Zave and Jackson (1997) is the form the proof should take. Gunter et al. (2000) interprets $W, S \vdash R$ proof in a predicate calculus setting, but without considering time; Hall and Rapanotti (2003) add time. The same authors, in Hall et al. (2006), show how even very weak notions of 'proof' can be used with solutions argued adequate, or *fit-for-purpose*, with respect to the stake-holders in the development. In this paper, with a similar abuse of terminology, we use *proof* to stand for any argument that argues the adequacy of $W, S \vdash R$.

Naively one might think that, if disaster has struck, there must be at least one witness against a 'proof' of $W, S \vdash R$ – a missed test case, for instance – so that the proof must simply have been incorrect. In each of the examples given earlier, however, proofs, in the form of safety cases, will have existed and so we conclude that there are other possible explanations for an accident, even given a 'proved' $W, S \vdash R$. It is these possibilities and their consequences that we explore in this paper.

The first possibility is that the designer(s) misunderstood the world W as W' and this led them to implement a solution S to a different problem, providing a proof only for $W', S \vdash R$ instead of for $W, S \vdash R$, i.e., not for the world in which S would actually be deployed. In this case, a proof exists, but not for the installed system. An incident can then be understood as stemming from the differences between W and W'. As an example, consider when W' ignores *aquaplaning* as a possible behaviour of a plane on a runway; in this

[2] There are other elements of the relationship described in Zave and Jackson (1997) but they are not relevant to our discussion.

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

case, $W = W' \cup \{aquaplaning\}$, and behaviours involving *aquaplaning* would not have been considered in the proof.

The second possibility is that the designer wrongly interpreted the requirement R as R' and this led them to again implement a solution S to a different problem, providing a proof only for $W, S \vdash R'$ instead of for $W, S \vdash R$. An incident can then be understood as stemming from the differences between R and R'. As an example, consider a device whose requirements omit to say that the interface should not allow the target to be coincident with the device. Then we have $R = R' \cup \{not\_coincident\}$, and the proof may miss initialisation after battery replacement.

The third possibility is that the designer correctly understood the world and requirements, and found S and a proof that $W, S \vdash R$ but, due to bugs in development, in the compiler, etc., they only managed to build S'. Examples of this form of error are manifold.

There are other possibilities due to operator misunderstandings during operation. Again, the operator can misunderstand W, R or S. The misunderstanding may, for instance, lead the operator to intervene when no intervention was necessary; in the worst case, the operator will intervene when intervention leads to an incident.

Problems and mishaps occur precisely because each of these possibilities are very likely to happen, not only singly but in concert. That it does not occur more often is testimony to safety engineering. The only systems in which it will not occur are those in which it is not possible to misunderstand W, R and S, i.e., toy problems.

In this section, we present a framework for the analysis of these possibilities. We will then consider the relationships between them, including the important case of when one error causes another, and another, and so on. Our framework also takes into account initialisation issues, such as those involved in the dead battery incident.

2.1. Interpreting WSR model as a behavioural model

A general safety-critical system will work by the system S sensing the world W through sensors and controlling it through actuators, both of which are in the world. During operation, the system will include feedback and operator control which we also consider part of the world. From this perspective, we may consider the world-system state at time t, at which a number of things can occur:

(1) a sensor could supply input to the system; additionally some system input could be supplied by the operator;
(2) an actuator could take output to the world; additionally the operator could receive some feedback from the system;
(3) some internal software step, i.e., processing that is unobservable by the operator, could be started by the system, perhaps using sensor input (at time t), and ending at t';
(4) the world will (possibly) start to change, by itself or influenced by actuator output (at time t), and ending at t''; this change may or may not be observable to the operator.

Behaviours of the system are then chains of such steps. To simplify the analysis, and without loss of generality, we will focus only on those t that occur at 'notable' events, such as sensor (or operator) input, or actuator output, or failure (or even disaster), and on steps that are delimited by such. We also assume, again without loss of generality, that (a) and

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

(b) do not occur together, i.e., that notable events are separate in time; this assumption allows us to abstract simply from time, and so we write:

(1) a "sensor" step as $W \rightarrow {}^{ws}S$, where *ws* signifies a world to system, i.e., a sensing, step;
(2) an "actuator" step as $S \rightarrow {}^{sw}W$, where *sw* signifies a system to world, i.e., an actuating, step;
(3) a "software" step as $S \rightarrow {}^{s}S$, where *s* signifies a system step;
(4) a "world" step as $W \rightarrow {}^{w}W$, where *w* signifies a world step

Fig. 1 illustrates the relationships between these steps.

2.2. Multiple views of the WSR model

Fig. 1 can be easily augmented with designer and operator views of the various steps, leading to Fig. 2. In the figure, there are three concentric squares: in the centre is the actual behaviour, that of Fig. 1; the outermost square corresponds to the operator's view (with world and system states subscripted by *o*); the innermost square corresponds to the designer's view (with world and system states subscripted by *d*).

In the figure, W_d is either:

- the actual state of the world W as it would be at some time t (written W^t), as viewed by the designer who considers it as holding at a time t_d, not necessarily equal to t (written $W_d^{t_d}$). In effect, the designer might get the wrong state at the right time, the right state at the wrong time, or the wrong state at the wrong time; or
- the lack of a designer view of the world state. In effect, the designer has not expected a notable event to have occurred.

Mutatis mutandis, each W_o represents the state of the world as perceived by the operator; S_d represents the designer's view (or lack thereof) of the system in operation, and similarly for S_o. Of course, in the perfect case, $W_d^{t_d} = W_o^{t_o} = W^t$, and $S_d^{t_d} = S_o^{t_o} = S^t$.

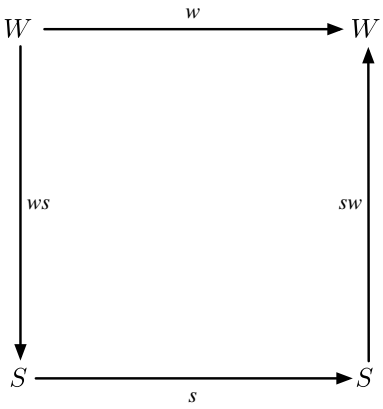

Fig. 1. Sensor, actuator, software and world steps, combined.

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

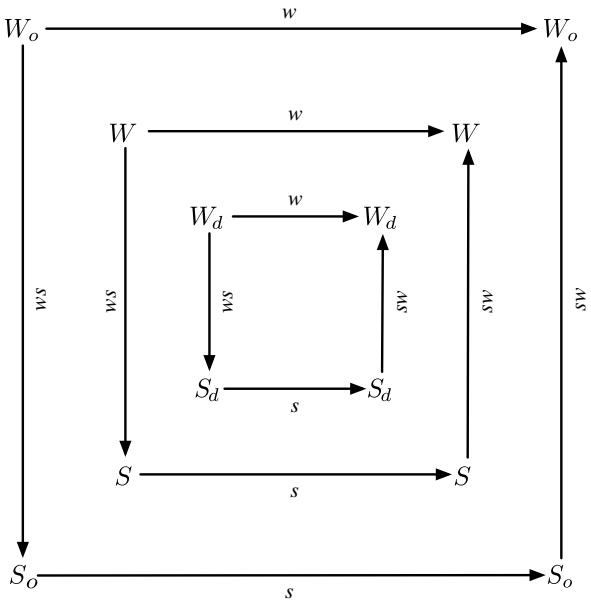

Fig. 2. The actual behaviour together with the operator's (outer square) and designer's (inner square) view.

The actual behaviour of the machine and world, as illustrated in Fig. 1, may include failure and other deviations. Being unpredicted necessarily indicates some divergence between the squares in Fig. 2 and, following (Silva, 2002), we focus not on the various views themselves but on the *discrepancies* between them.

What we mean by this is shown in Fig. 3. From Fig. 2 we add four *X-expressions* and eight *Δ-expressions*. Beginning with the *X*-expressions, representing discrepancies between states (of which there are four distinct ones):

X_o^w reflects any discrepancies between the operator's view of the state of the world and the state the world is actually in.
X_o^s reflects any discrepancies between the operator's view of the state of the system, and the state the system is actually in.
X_d^w reflects any discrepancies between the designer's view of the state of the world, and the state the world would actually be in during operation.
X_d^s reflects the discrepancies between the designer's view of the state of the system, and the state the system would actually be in during operation.

The *Δ*-expressions represent discrepancies between steps (of which there are eight), four of the operator and four of the designer. For the operator during operation:

Δ_o^w The operator misconstrues an environmental step; in other words, the operator misinterprets world events, even when those events are nothing extraordinary or unexpected;
Δ_o^{sw} the operator misconstrues an actuator step: the operator is confused about the outputs from the machine (including feedback); however, the output in itself is correct.

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

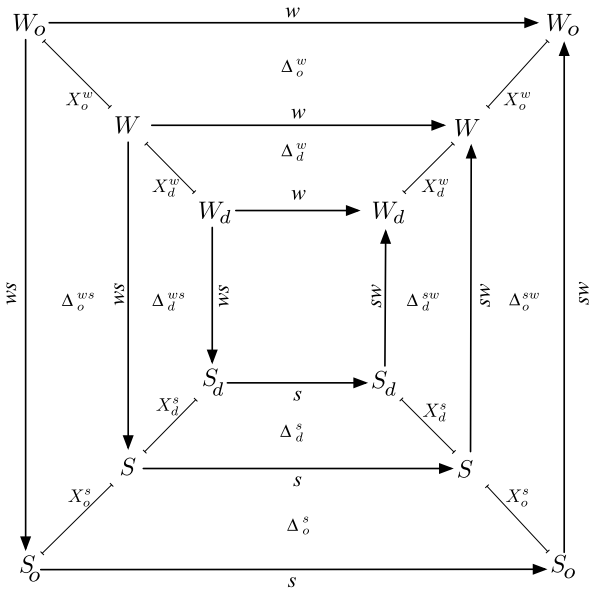

Fig. 3. The basis of our analyses, in which the *discrepancies* between views of states and steps is represented.

Δ_o^s the operator misconstrues the sensor/actuator linkage or relationship; the operator misunderstands the software function or behaviour; however, the system behaves correctly.

Δ_o^{ws} the operator misconstrues a sensor step; the operator is confused about the inputs provided to the machine, including their own, even when those inputs lie within the expected behaviour.

For the designer at design time:

Δ_d^w the designer misconstrued an environmental step; world events happen that were unexpected at design time;

Δ_d^{sw} the designer misconstrued an actuator step; outputs, unexpected at design time, confuse the operator.

Δ_d^s the designer misconstrued the sensor/actuator linkage; the software step did something wrong and this was not accounted for in the design;

Δ_d^{ws} the designer misconstrued a sensor step, perhaps by the operator; input to the machine is wrong from the designer's point of view.

3. Conceptualising mishaps as discrepancies

As is clear from Fig. 3, any discrepancy between a designer and/or an operator view of a state, i.e., one of the *X*s, must be arrived at either from a previous (i.e., accumulated) discrepancy between steps, starting from some first Δ located at the root cause of the discrepancy, or from a discrepant view of the initial state. The main hypothesis of this work is

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

that all undesirable states are describable as *X*s, arrived at from a root *Δ*, or from a discrepant view of the initial state. Should this hypothesis hold, it grants both systematic forwards and backwards analysis of any discrepancy between an operator's (or designer's) view of a step (or sequence of steps) and the actual step(s) that took place.

There are two parts to our presentation: the first is an analysis of the interactions that occur between discrepancies; the second is a catalogue of the discrepancies that can occur.

3.1. The interaction of Δs and Xs

For the analysis of undesirable events (accidents, incidents, mishaps) it is important to look at the dynamics of the interactions between different undesirable *Δ*'s and *X*'s. Understanding these dynamics is the basis for a proper understanding of how discrepancies accumulate, possibly leading to an accident; in particular, how those of the designer influence the operator. In Fig. 4 we show the cause–consequence view of Fig. 3 which helps in the analysis of the dynamics.

The observation which forms the basis of the diagram is that, given any discrepancy in the designer's or the operator's view, i.e., for any *X*, we can trace its origins back either:

(1) to a correctly perceived previous state, made discrepant by a *Δ* step;
(2) to a discrepant previous state, one of the *X*s that cascades into this one; or
(3) to a discrepant view of the initial state (not shown in the figure).

The importance of Fig. 4 is that, through it, we clearly see the "paths" that connect designer's discrepancies with those of the operator, represented as dot-dashed lines in the figure. These are precisely the paths by which mistakes at design time lead to problems at operation time, also notable in that they are one-way: because of their temporal distance, operator mistakes do not propagate their consequences to design.

Working from the figure, it is possible to generate complete, backwards and forwards analyses that, given a particular discrepancy, trace its possible causes and/or consequences. In the remainder of this section we populate the steps involved by focussing on each of the *X*s in turn.

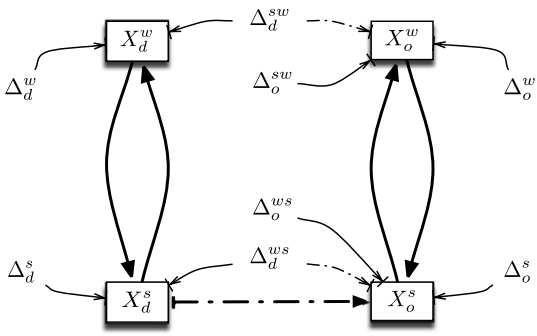

Fig. 4. Schema of the relationship between *Δ*s and *X*s. Essentially, *Δ*s correspond to questions of the form "has something failed since the most recent notable event?", whereas the *X*s correspond to questions of the form "has the problem been inherited from a previous state?".

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

It is also interesting to note that bad feedback design and bad input design are included in this classification which would contribute, respectively, to increase the *gulf of evaluation* and the *gulf of execution*, in Norman's "three model" framework (Norman, 2002), similarly based on a separation of view.

3.2. Analysis of the X^s_d situation

A state is labelled as X^s_d when there are "design" errors in the system state with respect to the requirements. The four immediate possible causes of X^s_d are:

- an immediately previous X^w_d discrepancy (in Fig. 4, follow the bold arrow back from X^s_d to X^w_d, i.e., in this case the problem did not start in the machine or in the software but came before from the designer's mistaken view of a previous world state cascading through to their design. For instance, that aquaplaning was not understood as a real-world phenomena meant that no account of it could be taken in a software controller that led to wheel braking;
- a Δ^s_d (in the figure, the light arrow with this label), i.e., the discrepancy is caused by a machine that does not perform its execution cycle as expected by the designer (even when the inputs operated correctly). Examples of this form include the effect of traditional programming errors, compiler introduced bugs, etc.
- a Δ^{ws}_d (etc.), i.e., something unexpected happened in sensing the world. For example, a button was pushed when it was supposed to be disabled, or a sensor gave unexpected input, *etc*.

Additionally, the X^s_d could be an incorrect view of the initial software state, such as can happen when the software contains uninitialised variables that take arbitrary values at run-time.

Note that, because these problems are at design time, it makes no sense to ask questions here about the role of the operator, other than as a source of behaviour to be accounted for through design, as there are no means for unexpected mistakes during operation to influence the system design.

3.3. Analysis of the X^s_o situation

A state is labelled as X^s_o when there are "operator" misconceptions regarding the system state in operation, with respect to its requirements. In effect, the operator's behavioural model of the system and the actual behaviour have diverged. Such confusion is closely related to the "interaction" quality of Perrow (1999). The immediate possible causes of X^s_o are:

- an immediately previous X^w_o discrepancy (bold arrow), i.e., in this case the problem did not start in the machine or in the software but was a consequence of the operator's incorrect view of the world cascading through to their current response. For example, suppose the operator wrongly thinks that the temperature is x when it is y. A machine programmed to respond differently to temperature x than y will then behave differently to the operator's expectations. This is closely related to "mode confusion", caused by differences between the actual mode of the machine and the mode as seen by the operator (Leveson et al., 1997).

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

- a X_d^s discrepancy (the dot-dashed arrow), i.e., in this case the confusion of the operator is due to discrepancies between the machine and the actual events. This case is particularly notable in that the discrepancies were introduced at design time. For instance, consider the case of a bug in the software leading the system to a state in which a temperature is calculated to be over some critical threshold so that the machine initiates emergency action, all to the surprise of the operator, who knows the temperature to be below the threshold and expects the machine to have correct information.
- due to Δ_o^s, i.e., the operator has misunderstood just the most recent software step. There are many reasons: perhaps there is some confusion, stress, or poor training with respect to the machine.
- a Δ_o^{ws}, i.e., the operator does not understand sensor input to the system; this time the misunderstanding comes from ignorance or lack of attention (or other non-technical reasons).
- a Δ_d^{ws}, i.e., the operator provides the expected information according to the current information and state of the system, but it was incorrect because of confusion, perhaps because of the interface. An alternative cause of operator confusion is not physical but procedural, or related to bad documentation. In the Cali aircraft accident (Leveson, 2004) there was a problem in the flight charts documentation that led the pilot to type a code that did not correspond to the flight path actually desired. The pilot typed 'R' in the FMS, with the intention of flying to *Rozo*, but 'R' was, in fact, the identifier for *Romeo* according to the Flight Management System; entering 'R' caused the plane to take a different path to that expected. So the system operated correctly, as did the operator, but the input provided was still incorrect.

Additionally, the X_o^s could be an incorrect view of the initial state of the system, perhaps caused by inexperience or lack of training of the operator.

3.4. Analysis of the X_d^w situation

This describes a state where there are differences between the designer's expectation of the world and how it actually behaves. From Fig. 4, there are three immediate possible causes (to which we add an initially discrepant view). There are two kinds of possible misunderstandings here that deserve to be analysed. In the first it is possible that the designer did not understand how the environment behaves, etc. In the second it is possible that, even though the designer understood the environment fully, a design error led the system to attempt to influence the environment in an undesirable way. So, the immediate possible causes of X_d^w are:

- an immediately previous X_d^s discrepancy, i.e., even when sensors and actuators were free from problems, a system action (or lack thereof) has occurred, the cause of which was rooted in design. Did the X_d^w happen because the designer misunderstood how the system updates itself? If so, it could be that the resulting actuation was incorrect: in the now classical chemical reactor example (Leveson, 2004), a water valve did not open because the designer assumed it would not be coincident with a request for maintenance.
- a Δ_d^w, i.e., the environment behaved unexpectedly; this includes the quaintly named 'Acts of God' as well as, perhaps, less understandable oversights by the designer. In this case it may be appropriate to ask how the designer's characterisation of the environ-

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

ment has been deficient, and/or which of the designer's assumptions were incorrect. Unexpected aquaplaning would fit here.
- due to Δ_d^{sw}, in which actuation behaviour deviates from that designed for. An example, might be a turbine blade, under the control of software, that is realigned incorrectly due to poor understanding of the control mechanism.

Additionally, this case covers an initially discrepant view of the state of the world leading, say, to an inconsistent model state in the software. Preventing such initial discrepancies is the role of the initialisation concern in the Problem Frames approach (Jackson, 2001).

3.5. Analysis of the X_o^w situation

Here, the operator is confused about the state of the world, i.e., there are discrepancies between the operator's understanding of the world state and what that state actually is. This confusion is related to the "coupling" quality of Perrow (1999). From Fig. 4, there are four immediate possible causes of X_o^w (to which we add an initially discrepant view). These are:

- due to a previous X_o^s, i.e., the operator was previously confused, and this has set expectations that have not been met by the world's behaviour. This is, for example, where the physical effects of mode confusion would become apparent.
- due to Δ_o^w, i.e., some notable event has occurred in the environment but this has not been perceived by the operator whose understanding, as a consequence, diverges from reality. For example, an operator that is inattentive may not have noticed that the temperature of a reactor vessel, indicated by some instrument, has increased to beyond some noteworthy point.
- due to a Δ_o^{sw}, i.e., the operator has, for instance, misread or misunderstood some indicator of the system state. We are careful to distinguish here between operator error, unmitigated by the actions of the designer, and the following case:
- due to a Δ_d^{sw}, i.e., in which *the operator's confusion is caused by the designer*. Of particular relevance is poor interface design, causing poor feedback in operation. In the Iranian Airbus accident (Neumann, 1994) poor feedback led the operator to fire on a civil aircraft. In this case, although it was the operator's actions that caused a catastrophic error, the assignment of responsibility is clearly with the designer, as the actual source of the error lies with the interface design.

Additionally, this case covers a discrepant view of the initial state of the world.

4. The conceptual model and the interaction analysis challenges

The elements of our framework form a basis for the detailed analysis of the outcomes of design and of operation and, importantly, their interaction. In the framework, then, we are able to address the interaction challenges of Section 1.1.

Overcoming the challenge of the designer: Using the framework, the designer can focus on the system, analysing its interactions with the environment and with the operator.

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

Undesirable interactions can be classified and traced backwards to causes, and forwards to effects.

Overcoming the challenge of the operator: Although the framework will be of limited applicability in the heat of operation, analysis under the framework can lead to the synthesis of better training materials, for instance, that properly focus on any difficult issues related to the display of information and to the control of the system.

Overcoming the challenge of the safety analyst: Under the framework, the focus of the analyst can simultaneously be on the operator, the system and the environment: the ''human path'' and the ''system path'' and their interactions can be considered together.

5. Implications of the model: overview and discussion

The purpose of this paper is not to introduce in detail how the conceptual model can be applied to practical systems, but to introduce the reference model itself. However, it is possible to sketch a range of possible applications and the kind of advantages provided.

First, we note that the approach is technology agnostic and is independent of any particular notation used to represent and reason about it. It is also independent of the lifecycle and can be used at different points: in system design, in training, for assessment, and for forensic analysis.

Second, the framework can be used to support other processes. Consider, for instance, hazard identification and analysis. Hazards, under our model, are seen as (accumulated) discrepancies from the intended. For hazard identification, a process such as the following could be beneficial: take a, perhaps preliminary, model of the system and consider scenarios or task descriptions. Map the tasks to steps, according to the *w*, *ws*, *s* and *sw*. For each step list the devices that come into play (sensors, actuators, buttons) and make a list of possible deviations, both from a designer's (*d*) and from an operator's (*o*) point of view. As deviations are identified, measures can be considered to avoid or control them.

Alternatively, when (generic) hazards are known, other support exists: for instance, by locating a hazard in Fig. 4 as *X*s, it is possible to trace backwards through causing discrepancies.

More generally, once a deviation has been identified and properly classified, using Fig. 4 as reference, two ''pure'' analysis strategies are possible:

- Forward analysis, with the aim to trace possible consequences: consider each discrepancy's next world, following the arrows forward, and decide if they are dangerous or not.
- Backward analysis, with the aim of tracing possible causes: consider a discrepancy in *w* and trace backwards towards causes, following the arrows backwards.

Of course, mixed strategies – combining forwards and backwards analyses – are also possible.

As a brief example of the framework's use for building a narrative of the events that took place in an accident, let us review the ''dead battery'' case under this framework, following a backwards analysis. First, as for any other accident, it happened in the world not in the system and at operation time so that the accident can be identified as an X_o^w situation, i.e., there was a discrepancy between the actual events taking place at operation time and the operators expectations. Following Fig. 4 backwards, we find that there are four

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

possible situations that lead to X_o^w, three Δs and one X. In this case, the actuators operated without failure, so no Δ_o^{sw} or Δ_d^{sw} happened. Also, the world "operated" as expected (no "acts of God", so no Δ_o^w). Hence, we arrive to a previous discrepant state, classifiable as X_o^s, that can, again, be analysed backwards. After discarding other possibilities (no failures involved, etc.), the most plausible analysis leads us to fix our attention on a X_d^s discrepant state, i.e., a discrepancy between the actual state of the system and what the designer expected. Indeed, the system did not behave as the designer expected, as it had incorrect GPS co-ordinates. This state is initial, due, perhaps, to the poor initialisation of variables, and analysis can end here: in this way, we have now a sequence of events properly labelled and classified, that could, for instance, be the starting point for designing countermeasures in a new device.

Forwards, or bi-directional, analyses also explore possible future (combined with past) events, which can be used to guide the design of countermeasures, this time before the fact. In this case, we can take each hazard identified at its corresponding X stage; then, we propose designs for possible remedies for previous Xs or that will prevent a causing Δ. In addition, we can consider designs for possible barriers aimed to minimise, or eliminate, propagation to the next and future Xs. Of course, trade-offs for the different design options will be necessary, with preventive measures being, in general, more desirable. This process could also provide design rationale for safety-related design decisions. For example, in the battery example, once an initialisation issue was found as the problem that caused an X_d^s, several measures can be considered. First, the initialisation issue itself could be avoided by forcing the operators to manually input new co-ordinates after every battery change. Additionally, for avoiding problems of the X_d^s kind, special checks can be performed in the software for the presence of dangerous values in the variables (like co-ordinates too close to the co-ordinates of the device itself). In addition, other measures can be put in place at subsequent stages in order to avoid the propagation to that of the operator, X_o^s, such as by providing warnings to the operators about the actual co-ordinates or displaying a map that shows where the weapon is targeted. There are further measures to avoid an operator incident, X_o^w, such as by keeping the operator at some distance, if possible, although we note that this non-preventive measure is aimed only to reduce damage. Of course, different measures at different stages are not mutually exclusive, with each and their combinations having different implications for the overall safety.

We note that other – more traditional – approaches help and can be helped by our model. Time-lines, event-trees, fault-trees, etc, can be constructed following a step-by-step schema inspired by Fig. 4. For instance, every step (or group of steps) forward in an event tree can be made to correspond to each of the *w*, *ws*, *s* and *sw* steps, with possible deviations included in the analysis. Incident and accident reporting techniques (such as CAE (Johnson, 2001)) can also be supported, as there is a strong need to structure and organise the information in incident reports. In this way, our conceptual model complements other proposals and provides a unified basis from which different proposals can start, as the concepts included are not tied to any particular notation or analysis technique.

Other applications of the conceptual model have been sketched, to be explored and explained in future papers. We are interested in the use of the Xs and Δs for communication and incident reporting: their taxonomic qualities can be very valuable in that setting, as they can help to clarify things or to structure the questions to ask in any search for information, in the manner suggested by Johnson (2002). The Problem-Oriented

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

framework of Hall et al. (2006) could make use of the conceptual model to extend their system synthesis techniques from design to design *and* operation, allowing iteration between requirements, implementation and, supported by the framework, deployment. Other potential uses include structuring the verification process, helping to elaborate detailed scenarios of use and discovering related problems, helping to bring coherence to the results of an analysis distributed through different analysts or performing historical analysis of the most frequent Δs or Xs, in order to identify recurrent weak points.

We realise that the approach proposed is a model of execution that supports microscopic-type root cause analysis (RCA) (Cacciabue, 2004) and does not dig deep into organisational, psychological or situational factors. According to Cacciabue (2004), RCA is part of accident investigation: only part, but unavoidable. RCA explains events, but more investigation is required to find organisational and contextual factors. Combination of our model with other models, like STAMP or ISAAC, can lead to this macroscopic-type analysis, and we intend to research this topic in the medium term.

6. Related work

The approach we have presented combines models of human and machine frailty together in a single framework so that accident sequences involving both human and machine ''error'' can be represented and reasoned about. We now discuss how previous approaches have tackled this problem, and consider their achievements. We have found it useful, for the purposes of our presentation, to classify these approaches into three groups, characterised by their view of from where in the system exceptional behaviour derives. The three groups are:

- *mental cycle approaches*: how a human (incorrectly) reasons and interacts with a system that behaves as expected, i.e., exceptions come from the 'mental cycle';
- *machine cycle approaches*: how a machine (incorrectly) interacts with a human that behaves as expected, i.e., exceptions are caused by the 'machine cycle'
- *hybrid approaches*: in which both human and machine may not behave as expected.

6.1. Mental cycle approaches

These approaches focus on how humans reason and perform *with a system that behaves as expected*. It could be said that these approaches are focused on behaviours as observed through the ''mental cycles'' of the operator. Models of human error and performance belong to this class. Clearly, with a focus on the human, such approaches can overlook the deficiencies of the technical system.[3]

In Norman (1990), for instance, the author states that feedback is needed to compensate for the problems associated with ''over-automation''. We would observe that although feedback is needed, it is not sufficient: consider the situation in which too much, or too little, feedback is given to the operator confusing them and, worse, leading them to wrong decisions. During design, an engineer with a safety perspective would need to con-

[3] We are not saying that such approaches do not have their good uses in other areas.

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

sider just these situations in order to overcome them or to mitigate their effects. As another example, Fields et al. (1999) contributes to the design of Air Traffic Control systems by focusing on non-technical problems, such as communication, without considering the possibility that technical problems also hamper communication and add to confusion. The proposal by Galliers et al. (1999), built upon a cognitive model by Norman (1986), again puts a strong focus only on the cognitive-side of operations with the aim of designing safe user interfaces.

Cacciabue, in his comprehensive book (Cacciabue, 2004), discusses "models of cognition", like Perception, Interpretation, Planning, Execution (PIPE), but focuses on the human-side almost exclusively with only loose integration with the system and interaction. Accident models discussed by Cacciabue centre mainly on the operator's mind and behaviour, not on the software. Some models lack important features: for instance, the CREAM method (based on the COCOM model (Cacciabue, 2004)) looks for problems in the cognitive functions of the operator, but this makes it difficult to consider complete accident sequences, as many of the events involved do not relate to cognitive functions.

6.2. Machine cycle approaches

Simply stated, all traditional techniques for safety-critical systems (FTA, HAZOP, FMECA, ETs, etc.) belong here, and are described in classic books, such as Storey (1996) and Leveson (1995). We assume some familiarity of the reader with these approaches, and so do not describe them further. We point out that these techniques, as originally conceived, try to establish sequences of system or machine-related events, working backwards, forwards or both. Again, clearly, when the focus is on the machine such approaches may overlook the deficiencies of the human. Sometimes, however, human-related events can be included in the analysis using these techniques (for example, "operator inattentive" can be included as a leave in an FTA tree). In that case, we would arrive at a special kind of hybrid approach, to be discussed in the next section.

6.3. Hybrid approaches

Hybrid approaches assume failure comes not from a single source, i.e., they consider that *both system and humans may not behave as expected* across interaction cycles. In addition, some approaches also consider the environment of the system. The importance of this perspective is not only that it is closer to reality but that it helps designers overcome the design challenge of analysing and mitigating undesirable operator-system cycles of events. Our approach is, for this reason, located amongst the hybrid approaches.

We claim that, for application to safety-critical systems, existing approaches in this third group have been largely unsatisfactory.

Chris Johnson analysed this problem (Johnson, 1998), and proposed the use of formal and semiformal notations to build time-lines that show human factor and system engineering events. His analysis led him to the conclusion that the mere integration of different notations is unsatisfactory, as the nature of system evidence and human factors is qualitatively different. We agree, and understand that there can not be a "notation" or a even a group thereof that is fit-for-purpose. This fact encouraged us to look to a conceptual framework/reference model into which current notations would fit and/or from which new notations will be developed.

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

Hybrid approaches are often *technique-fusion* or *model-fusion* based, i.e., they are an eclectic approach based on a fusion between techniques or between their underlying models. The techniques tend to be traditional system-oriented approaches to safety (HAZOP, FMEA, *etc.*) combined with human-oriented approaches, like Human Reliability Analysis (HRA). Currently, they consider human factors and technical factors but, even though hybrid, deal with them separately or in a loosely coupled manner. Examples are (Taylor, 1994; Fenelon et al., 1994) or the ISAAC framework (Cacciabue, 2004). These approaches tend to use separate search branches, or paths, for human and technical events when, in fact, human and technical events especially at operation time are, as we have argued, strongly interdependent.

Another group of hybrid approaches arises from the combination of traditional techniques in new ways. For instance, SAM (Wilson and McDermid, 1995) aims to provide a set of consistency rules that should hold between techniques. Its goal was to overcome the problem of lack of integration of different techniques in a safety case, but the underlying model provided was an Entity-Relationship model that aimed to unify the use of terminology (system, component, risk, condition, severity, etc.) among different techniques, not to provide an underlying explanatory model of how different events interact. The HAZAPS approach (Broomfield and Chung, 1997) proposed a 4-stage process based on the identification of critical subsystems and keywords, making use of (modified) event diagrams and HAZOP-like guide words. The goal is to identify recommendations for enhancing the safety of a system. However the process is not systematic, with some analysis steps being ad hoc. Similarly, the Dynamic Flowgraph Methodology (Garrett and Apostolakis, 2002) is an enhancement of traditional FTA that includes the human operator as an external agent; however, interface and human–computer interaction issues are not given special attention. Another approach, inheriting the goals and principles of SAM, is HiP-HOPS (Papadopoulos et al., 2001) which modifies, automates and integrates the techniques of FFA, FMEA and FTA. Again, it can not be considered as an approach that stems from an underlying model of human–computer–environment interaction and its range of applicability is restricted to electromechanical systems with limited human interaction. A recent development by Bush (2005) applies HAZOP guide words to models developed in the i^* notation (Yu, 1997). Again, at least at this stage of the research, the process is ad hoc and closely tied to i^*. Bush's technique applies the guide words to safety goals that have already been identified and so may help in the analysis of existing goals, but it can not uncover new goals.

Another relevant hybrid method is ADREP, discussed in Cacciabue (2004). The method ADREP is based on the SHELL structure (Cacciabue, 2004, p. 83) composed of Software (understood not as computer software but as a set of rules that personnel follow), Hardware, Environment and Liveware (people). It focuses on communication problems between those elements of the model, paying less attention to cognitive factors. This method presents an accident as a time-sequence of visible events, but provides no logical connection between events. Consequently, a logical relationship between system and operator events is difficult to trace.

Leveson has recently proposed a new accident model, named System-Theoretic Accident Model and Processes (STAMP) (Leveson, 2004), that takes into account the organisational and design issues that play an important role in system accidents. STAMP, originally, focused on forensic analysis and now is being extended for finding hazards and for designing countermeasures. One of the goals of the STAMP model is to call atten-

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

tion to root causes at some spatial or temporal distance from the operation of the system, but which are deleterious during operation. In this way, an analyst that uses STAMP is forced to look beyond the particular events that took place, now seen as consequences of (perhaps) high-level decisions or design issues. This is a remarkable goal. Unsurprisingly, for proper analysis, the actual events are still needed and their identification and analysis is still unavoidable. For this purpose, STAMP includes "control flaws", a general classification that does not take into account the integration of human and technical factors or events, and the close interrelation between flaws of those different natures. Additionally, some interface-related issues may be overlooked; for instance, possible dangerous misunderstandings by the operator about how an actuator performs its job, or how the software actually transforms its inputs to outputs.

7. Conclusion

Safety is an holistic issue, deriving from environment/human/system factors. From our analysis of the state of the art, no single current approach has provided a full account of all factors and a systematic approach to their interactions. We believe that the current ad hoc approaches are due, in part, to a lack of a strong conceptual foundation the issue being, essentially, that there is no single model that can underpin combinations of tools, methods and techniques. In this paper we aimed to provide an approach, based on the *WSR* model (Gunter et al., 2000), that provides such a conceptual framework. Note, however, that the provision of a framework cannot, per se automatically join techniques nor correct any deficiencies in other approaches – that must remain the aim of future work.

Here we have presented a notation- and method-independent reference model and derived from it a conceptual framework for the analysis of undesirable interactions between events at operation time and at design time. We hypothesise that our reference model brings with it formal, step-by-step, structure to discover, understand and describe human–computer–environment hazards. Also, it is compatible with previous notations, techniques and models, as it helps to provide some structure to analysis that, in many cases, are carried out in an ad hoc fashion. Putting this reference model as the foundation, we intend to develop different methods and models useful at different points of the lifecycle of safety-critical systems with a human component.

Acknowledgements

The authors would like to thank their colleagues in their respective institutions, especially Lucia Rapanotti, CRC, as well as the Royal Academy of Engineering and the *Universidad Politécnica de Madrid* for the funding under which this work was begun and has continued. The comments of the anonymous reviewers have also benefited the paper greatly.

References

Broomfield, E.J., Chung, P.W.H., 1997. Safety assessment and the software requirements specification. Reliability Engineering & System Safety 55 (3), 295–309.

Bush, D., 2005. Modelling support for early identification of safety requirements. In: Iee, E. (Ed.), 4th International Workshop on Requirements for High Assurance Systems (RHAS'05, Paris), August 2005.

Cacciabue, P.C., 2000. Human factors impact on risk analysis of complex systems. Journal of Hazardous Materials 71 (1–3), 101–116.

Cacciabue, P.C., 2004. Guide to Applying Human Factors Methods. Springer.

Edwards, W., 1962. Dynamic decision theory and probabilistic information processing. Human Factors 4, 59–73.

Fenelon, P., McDermid, J., Nicholson, M., Pumfrey, D., 1994. Towards integrated safety analysis and design. Applied Computing Review 2 (1), 21–32.

Fields, R., Paternó, F., Santoro, C., Tahmassebi, S., 1999. Comparing design options for allocating communication media in cooperative safety-critical contexts: a method and a case study. ACM Transactions on Computer–Human Interactions 6 (4), 370–398.

Galliers, J., Sutcliffe, A., Minocha, S., 1999. An impact analysis method for safety-critical user interface design. ACM Transactions on Computer–Human Interactions 6 (4), 341–369.

Garrett, C.J., Apostolakis, G.E., 2002. Automated hazard analysis of digital control systems. Reliability Engineering & System Safety 77 (1), 1–17.

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

Hall, J.G., Rapanotti, L., 2003. A reference model for requirements *engineering*. Proceedings of the 11th IEEE International Conference on Requirements Engineering (RE 2003). IEEE Computer Society Press, pp. 181–187.

Hall, J.G., Rapanotti, L., Jackson, M., 2006. Problem Oriented Software Engineering. Technical Report 2006/10, Centre for Research in Computing, The Open University.

Jackson, M., 2001. Problem Frames: Analyzing and Structuring Software Development Problems. Addison-Wesley.

Johnson, C., 1998. Representing the impact of time on human error and systems failure. Interacting with Computers 11 (1), 53–86.

Johnson, C., 2001. A case study in the integration of accident reports and constructive design documents. Reliability Engineering & System Safety 71 (3), 311–326.

Johnson, C., 2002. Forensic software engineering: are software failures symptomatic of systemic problems? Safety Science 40 (9), 835–847.

Johnson, C., 2002. Software tools to support incident reporting in safety-critical systems. Safety Science 40 (9), 765–780.

Johnson, C., 2003. A brief overview of causal analysis techniques for electrical, electronic or programmable electronic systems. Technical report, Department of Computing Science, University of Glasgow, Glasgow, Scotland.

Leveson, N., 1995. Safeware: System Safety and Computers. Addison-Wesley Professional.

Leveson, N., Pinnell, L.D., Sandys, S.D., Koga, S., Reese, J.D., 1997. Analyzing software specifications for mode confusion potential, In: Proceedings of the Workshop on Human Error and System Development. Glasgow, March.

Leveson, N., 2004. A new accident model for engineering safer systems. Safety Science 42 (4), 237–270.

Neumann, P.G., 1994. Computer-Related Risks. Addison-Wesley Professional.

Norman, D.A., 1986. Cognitive engineering. In: Norman, D.A., Draper, S.W. (Eds.), User Centered System Design: New Perspectives on Human–Computer Interaction. Erlbaum, Hillsdale, NJ, pp. 31–61.

Norman, D.A., 1990. The problem of automation: inappropriate feedback and interaction, not over-automation. In: Broadbent, D.E., Baddeley, A., Reason, J.T. (Eds.), Human Factors in Hazardous Situations. Oxford University Press, pp. 585–593.

Norman, D.A., 2002. The Design of Everyday Things. Basic Books.

Papadopoulos, Y., McDermid, J., Sasse, R., Heiner, G., 2001. Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure. Reliability Engineering & System Safety 71 (3), 229–247.

Perrow, C., 1999. Normal Accidents. Princeton University Press.

Reason, J., 1987. The psychology of mistakes: a brief review of planning failures. In: Rasmussen, J., Duncan, K., Leplat, J. (Eds.), New Technology and Human Error. John Wiley & Sons, New York.

Silva, A., 2002. Requirements, domain and specifications: a viewpoint-based approach to requirements engineering. In: Proceedings of the International Conference on Software Engineering 2002 (ICSE 2002).

Storey, N., 1996. Safety Critical Computer Systems. Addison Wesley.

Taylor, J.R., 1994. Developing safety cases for command and control systems. In: Redmill, F., Anderson, T. (Eds.), Technology and Assessment of Safety-Critical Systems. Springer-Verlag, pp. 69–781.

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001

Wilson, S.P., McDermid, J.A., 1995. Integrated analysis of complex safety critical systems. Comput. J. 38 (10), 765–776.

Yu, E.S., 1997. Towards modelling and reasoning support for early-phase requirements engineering. In: International Symposium on Requirements Engineering, Annapolis, MD, pp. 226–235.

Zave, P., Jackson, M., 1997. Four dark corners of requirements engineering. ACM Transactions on Software Engineering and Methodology 6 (1), 1–30.

Please cite this article in press as: Hall, J.G., Silva, A., A conceptual model for the analysis of mishaps ..., Safety Sci. (2006), doi:10.1016/j.ssci.2006.11.001