4

A Safety Engineering Perspective

Derek Fowler and Ronald Pierce
JDF Consultancy LLP
UK

1. Introduction

Safety is a viewpoint. By this we mean that safety is not in itself an attribute of a system but is a property that depends on other attributes and on the context in which the system is used. The question that arises (and which we will attempt to answer in some detail) is which attributes of a system determine whether it is safe or not, in its context of use.

Throughout this chapter, the term *system* is used in the widest sense - ie it includes not just the technical elements (equipment) but also all other elements - eg the human-operator and operational procedures - that necessarily make up the complete, end-to-end system.

We will start by attempting to dispel what seems to be a not-infrequent misconception (also reflected in some safety standards) that safety is mainly dependent on reliability (and / or integrity, depending on one's definition of the terms - see section 3.1 below). This we feel is important for those readers who may have had some previous exposure to areas of safety engineering in which this view is held and will lead to the inescapable conclusion that we need a broader view of system safety than is sometimes taken.

Next we will establish some basic safety concepts firstly by defining key terms, and then by considering two distinct categories of safety-related system and seeing how the system properties determine safety in each case.

Finally, and for the most part of this Chapter, we will explain how the broader approach to safety works and show that it is closely linked with (not 'special and different' from) systems engineering in general.

2. "Safety is reliability" – Dispelling a myth

[Leveson, 2001] in a review of major software-related accidents and the implication for software reliability, presents compelling evidence that software reliability had never been the cause of such disasters - on the contrary, in every case investigated, the software had performed in exactly the manner that it was designed to. The problem was that the software was designed to do the wrong thing for the circumstances under which it "failed" (or, as in the case of Ariane V, for example) was used for a purpose (ie in a context - see above) different from that for which it was originally designed. Professor Leveson quite rightly, therefore, poses the question as to why, in most software safety standards, so much emphasis is placed on processes to improve software reliability whilst not ensuring also that

the resulting systems actually perform the intended function – ie allowing them to be what one might call "reliably unsafe". This same misconception is prevalent also at the system level in, for example, European Air Traffic Management (ATM) - see [Fowler, 2007].

We can illustrate the problem by considering the simple, everyday example of a car airbag for which, for the sake of this discussion, we wish to make a case that it would be safe.

If we were to simply follow a failure-based process - ie focus on how reliable the airbag needs to be in order to be 'safe' - we would start (at the wrong point, as we will see shortly) by identifying the hazards[1] presented by the airbag. Such hazards are those caused by the airbag's two main failure modes: failure to operate when required, and operating when not required. We would then use a risk classification scheme to derive safety requirements that specify the maximum frequency with which those failures could be allowed to occur and from that we would deduce more detailed safety requirements which limit the frequency with which the causes of the hazards could be allowed to occur.

Even if the results were valid, they would lead us only to:

- an understanding of how reliable the airbag needs to be - so that it operates when required; this would, however, not give any assurance that, when it did operate, the airbag would actually protect the front-seat occupants from death or serious injury in the event of a collision, and
- the totally **irrational** conclusion that putting an airbag in a car would only increase the risk of death or serious injury to the front-seat occupants, because of the finite (albeit small) possibility that it would operate when not intended to!

Of course, what is missing is any evidence of a positive safety contribution from the airbag - only the possibility of actually being killed / seriously injured by it - without which we would have no case for fitting one.

If instead we were to take a more **rational** view, we would start from the position that in the event of, say, a head-on collision without an airbag there is a very high risk of death or serious injury to the driver (and other front-seat occupant(s)) of a car. This risk we can call *pre-existing* because, by definition, it is inherent in driving and has nothing whatsoever to do with the airbag – indeed it is to mitigate this risk that we are intending to fit the airbag in the first place. So, the more rational approach would be to:

- firstly assess how effective the airbag would be when it did work – ie by how much the *pre-existing* risk from driving would be reduced by the airbag - and what properties of the airbag determine the amount of this reduction; and then
- assess the *system-generated* risk, induced by airbag failure.

Thus, given the correct set of functional properties – eg shape, location, strength, compressibility, sensitivity to 'g' forces, speed of deployment etc – as well as adequate reliability and integrity, our safety case should show that the airbag would make a positive contribution to the reduction in the identified pre-existing risk that is very much greater than the system-generated risk due to airbag failure. This would be a much more balanced, and rational conclusion than what emerged above from considering only airbag failure.

[1] A state of a system that could lead to an accident - see section 3.1.

3. System safety – Basic concepts

3.1 Definitions

This section defines the safety terms that are used in the rest of this chapter. In most cases, as there is no single, universally accepted definition, the ones given below have been adapted from those in Part 4 of international functional-safety standard IEC 61508 [IEC, 2010] and, if not actually used by, should at least be understood in, any safety-related sector.

- *Harm* death or serious physical injury or damage to the health of people, or serious damage to property or the environment
- *Hazard* a situation, state or event that may result in harm[2]
- *Risk* combination of the frequency of occurrence and severity of harm
- *Safety* freedom from unacceptable risk
- *Reliability* the ability of a system / element to provide a long period of continuous delivery of an intended service (or function) without failure
- *Integrity* the ability of a system, under all stated conditions, to provide all the services (or functions) required by the users, with no unintended or un-commanded services (or functions)

Failure termination of the ability of a functional unit to provide a required function or operation of a functional unit in any way other than as required

3.2 Risk acceptability

3.2.1 The ALARP principle

Risk may, in extremis, be either so great that it would be intolerable under any circumstances or so small as to be insignificant and therefore may be discounted altogether. In practice, however, risk will usually fall somewhere between these two extremes and the ALARP principle requires that any risk shall be reduced to a level that is as low as reasonably practicable, bearing in mind two factors: the benefits resulting from its acceptance, and the costs of any further reduction. ALARP is described in more detail in IEC 61508 [IEC, 2010], Part 5, Annex C; other standards and practices use different acronyms such as ALARA (USA) and SFAIRP (Aus). In the UK, the ALARP principle has a specific legal connotation and expert advice should be sought before applying it! [Ladkin, 2008].

A practical way of specifying what is *tolerable* risk, and in some cases applying the ALARP principle, either qualitatively or quantitatively, is the so-called Risk Classification Scheme (also known as a Hazard-Risk Index).

3.2.2 Risk Classification Schemes

Risk Classification Schemes (RCSs) are used in a number of industry sectors. Their form is as variable as their usage but a typical example, from the ATM sector [EUROCONTROL 2010], is shown in Figure 1.

[2] Whether or not a hazardous event results in harm depends on whether people, property or the environment are exposed to the consequence of the hazardous event and, in the case of harm to people, whether any such exposed people can escape the consequences of the event after it has occurred

An RCS is typically set out as a matrix, in which the severity of possible outcomes is mapped against the frequency with which the outcomes might occur.

		Severity Class			
Probability per flt hr		1	2	3	4
Frequent	$P>10^{-4}$	A	A	A	B
Probable	$P<10^{-4}$	A	A	A	C
Occasional	$P<10^{-5}$	A	A	B	D
Remote	$P<10^{-6}$	A	B	C	D
Improbable	$P<10^{-7}$	A	C	D	D
Extremely Improbable	$P<10^{-8}$	B	D	D	D

Fig. 1. A Risk Classification Scheme

In this ATM example, the **severity** of outcome ranges from Class 1 (an accident involving death and/or serious injury[3]) to Class 4 (the lowest level of safety-significant incident) and, in practice, would be explained by detailed descriptions and illustrative examples. The **frequency** of outcome is shown both qualitatively and quantitatively, as the probability per flight hour (or per operating hour). The grid is populated with the tolerability / acceptability of the risk and, in this example, includes the ALARP principle[4] as follows:

- Risk Class A is defined as intolerable
- Risk Class B is tolerable only if risk reduction is impracticable or cost grossly disproportionate to improvement
- Risk Class C is tolerable if cost of risk reduction would exceed the benefit of improvement
- Risk Class D is defined as broadly acceptable

An RCS should be tailored to the purpose and function of the system or service concerned and the risks and benefits involved. It would normally be published in the Safety Management System for the organisation responsible for the operation, and be approved by the relevant safety-regulatory authority. It can be used in one of two ways:

- for safety monitoring of an on-going operation (see section 6.5 below) - in this case the achieved risk can be compared with what is tolerable / acceptable according to the RCS, and / or
- for *a priori* safety assessment - in this case the severity of each potential outcome is assessed and the required maximum frequency of occurrence, in order to achieve an acceptable (or at least tolerable) level of risk, is obtained from the RCS.

[3] In European ATM, it is not usual practice for accidents to be 'graded' by the number of people killed and/or seriously injured.

[4] If the ALARP principle were not applied to the RCS, the red boxes (labelled 'A') might remain the same as in Figure 2 but the rest of the grid would show only what was tolerable.

One of the main attractions of RCSs is that they are relatively simple to use - and therein lies a potential problem, unless each user it careful to check the following:

- at what level in the system hierarchy, and within what scope, the values apply
- where the probability / frequency values used in the scheme came from and whether they are appropriate and (still) valid
- how the aggregate risk can be deduced from analysis of individual hazards, in restricted segments of the total system
- what allowance needs to be made for *pre-existing* risk - see also section 3.3.3 below.

The significance of some, if not all, of these 'health warnings' should become more apparent in the subsequent sections of this chapter.

3.3 Safety-related systems and their properties

3.3.1 Safety-related systems – General

Consider the two types of safety-related system (SRS) shown in Figure 2. Case (a) is a system - say, a complete nuclear power plant - which simply provides a service into its operational environment. Because the service in the case of a nuclear power plant is the provision of electrical power then, from a purely safety viewpoint, we do not care whether the service is provided or not. What we do care about are the hazards (eg radiation leakage), and the related level of risk, that a failure internal to the system might present to its operational environment (and to the people therein).

Case (b) is quite different. Here we have, first of all, a set of hazards that already exist in the operational environment. If, for example, the System was our car airbag (see section 2 above) then these hazards (and associated risks) would be those (*pre-existing*) hazards / risks inherent in driving a car, and the operational environment (from the airbag's perspective) would be the whole car.

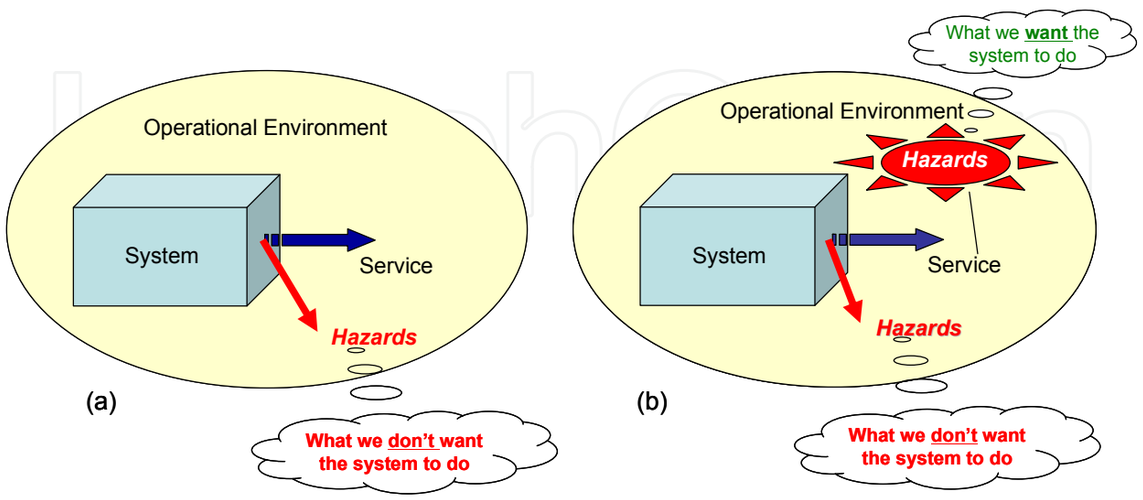

Fig. 2. Two types of Safety related System

As we will see in more detail in section 3.3.3 below, what is very important about Figure 2 is that for case (b) the mitigation of *pre-existing* hazards / risks (ie what we want the system to do) and the inevitable introduction of *system-generated* hazards / risks (ie what we don't want the system to do) depend on entirely different properties of the system.

Before that, however, we will introduce IEC 61508 [IEC, 2010], probably the most widely accepted, international standard on the functional safety of systems.

3.3.2 IEC 61508

IEC 61508 has a particular model of how SRSs influence the real world that is based on the concept of the *Equipment Under Control* (EUC) which itself is regarded as the hazard-creating system[5] and for which SRS are designed in order to mitigate those hazards [Pierce & Fowler, 2010]. Since IEC 61508 is a generic standard, to be adapted for application to a wide range of specific industry sectors, it has no particular view on the nature of the EUC, which could be a nuclear reactor, a chemical plant, an oil rig, a train, a car, or an aircraft etc[6].

The standard then defines two types of SRS that are intended to mitigate the hazards and risks associated with the EUC:

- *Control Systems* (eg a railway signalling system) which provide *Safety Functions* that are designed to maintain continuously a tolerable level of risk for the EUC, and
- *Protection Systems* (eg an automatic train protection (ATP) system or car airbag) which provide *Safety Functions* that are designed to intervene when they detect a hazardous state developing within the EUC and/or its Control System(s), and put the EUC / its Control System(s) into a safe, or at least safer, state[7].

As far as a Control System is concerned, the hazards and risks associated with its EUC are clearly *pre-existing*, since they are caused by the latter not the former. Similarly, the hazards and risks associated with the combination of an EUC and its Control System(s) are pre-existing as far as a Protection System is concerned.

With this concept, an SRS (Control and/or Protection system) is put in place to reduce the pre-existing risks to an acceptable level. IEC 61508 refers to this as *Necessary Risk Reduction* but does not actually stipulate what is "acceptable", this being left to local or national considerations, including legal frameworks, for the applicable industry sector.

As we will see in section 3.3.3 below, safety integrity requirements on Control Systems are usually expressed as probability of failure per operating hour, whereas for Protection Systems they are usually expressed as probability of failure on demand. In either case, the target probability will of course depend on the level of the pre-existing risk. The

[5] Equivalent to Figure 2 case (a).

[6] In these examples, the EUC is very tangible and for these it is probably a better term than the equivalent term "operational environment" used in Figure 3(b). However, in some cases - eg air traffic management and a railway level crossing (see section 4) - the EUC is much less tangible and "operational environment" might be better. Whatever term is used, the principles are exactly the same and the concept of pre-existing risk is paramount!

[7] Eg, an ATP system is designed to stop a train if it passes a signal at danger.

objective of the SRS for Control and Protection systems is *risk control* and *risk reduction* respectively[8].

In both cases, IEC 61508 is quite clear that the safety functional requirements (specifying functionality and performance of the Safety Functions) must be completely and correctly identified before the SRS can be designed. This requires hazard and risk analysis of the EUC not (initially at least) hazard and risk analysis of the SRS(s) themselves. Once the safety functionality and performance requirements of the Safety Functions have been identified, the tolerable failure rates of the Safety Functions can then be identified, and the Safety Integrity Level (SIL) for each Safety Function established[9].

3.3.3 Safety properties

We can build on our simple example of a car airbag to explain more generically, and develop, the above principles since it can readily be seen that an airbag fits Figure 2 case (b), and the IEC concept of a Protection System, very well. Figure 3 shows the risk, in the Operational Environment (or EUC), with and without an SRS – ie R_U and R_A respectively. As we saw for the airbag in section 2 above, the safety case for the SRS depends on its making a (much) bigger positive contribution to safety when operating as intended (ie the *success* case as represented by the green, right-to-left arrow) than any negative contribution caused by its failure or incorrect / spurious operation (ie the *failure* case as represented by the solid red, left-to-right arrow).

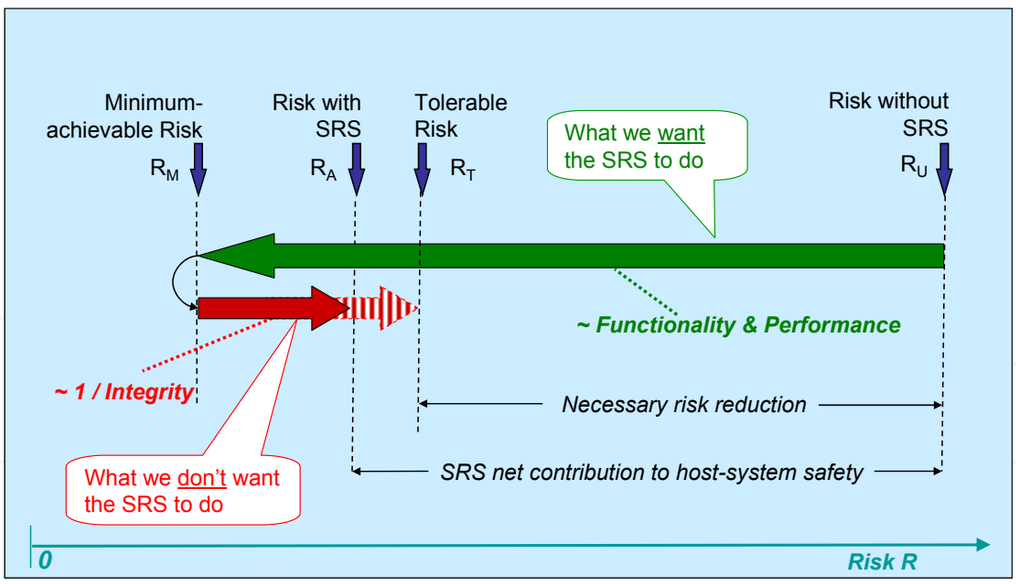

Fig. 3. Risk Graph for a Safety-related System

[8] Version 2 of IEC 61508, issued in 2010, is much clearer about the application of risk reduction to protection systems and risk control to continuously-operating control systems, than was the earlier (2000) version.

[9] This is exactly what we said for the airbag example in section 2, but is not always followed in some industry sectors!

There are a number of very important points to note about this diagram:

- R_U has nothing to do with the SRS – ie it is the *pre-existing* risk, as above
- R_M is the theoretical minimum risk that would exist in the complete absence of failure / spurious operation of the SRS – it is not zero, because there usually are some accident scenarios for which an SRS cannot provide mitigation[10]
- since R_M is defined as the risk in the absence of failure, it must be determined only by the *functionality & performance* of the SRS, as explained in section 2 above
- the risk increase R_A-R_M is caused entirely by failure / spurious operation of the SRS - thus it is the *system-generated* risk and is determined primarily[11] by the *reliability & integrity* of the SRS
- the safety case for the SRS is based on showing that $R_A << R_U$
- if we now introduce R_T, the maximum tolerable level of risk, then an interesting conclusion emerges: given that R_T is fixed (eg by a regulatory body), then the maximum tolerable failure rate of the SRS - ie a function of the length of the extended red (l-r) arrow (R_T-R_M) - depends on the length of the green (r-l) arrow (R_U-R_M); in other words, the tolerable failure rate depends on how successful the SRS is in reducing the pre-existing risk in the first place
- overall, R_U-R_T fits the IEC 61508 definition of Necessary Risk Reduction
- if, as we desire, R_A-$R_M << R_U$-R_M, then the overall risk actually achieved (ie R_A) is much more sensitive to changes in the length of the green (r-l) arrow (ie to changes in functionality and performance) than to proportionate changes in the length of the red (l-r) arrow (ie to changes in reliability & integrity).

We can also see from Figure 3 that in the limit that R_M approaches R_T, so the integrity required of the SRS approaches infinity! This raises further important questions regarding the origins and use of traditional risk-classification schemes, which are often are based entirely on R_T and do not take any account of R_M in setting tolerable failure rates for a system. As we saw in section 3.2.2 above, RCSs generally model only the system's negative effects on safety, not its positive contributions and, therefore, to get a more complete picture of where risks lie in a system we need to turn to more sophisticated forms of risk modelling.

3.3.4 Risk modelling

One of the systems engineering techniques that is commonly used for the static modelling of risk in a safety assessment is Fault Tree Analysis (FTA) [IEC, 2006b]. This is illustrated, for a very basic Protection System, in Figure 4.

An accident would occur if one, or both, of two conditions occurs, as follows:

- firstly, the (pre-existing) hazard occurs and the consequences of the hazard are not mitigated; in this case, if the hazard were never mitigated, then the accident rate would be the same as the hazard occurrence rate – ie the hazard occurrence rate would be the

[10] Eg for an airbag these include fire, or being hit from behind by a vehicle with high relative velocity.
[11] The word "primarily" is used here because (as is more generally the case) it is may be possible to provide additional functionality to mitigate some of the causes and / consequences of system-generated hazards .

pre-existing risk (R_U) defined in Figure 3. The situation that the hazard is not mitigated could arise because Protection System either: operates but is not effective; or fails to operate.

- or secondly, the Protection System operates spuriously (eg when it is not supposed to, or in a way different from what was required) and the consequences of this (system-generated) hazard are not mitigated.

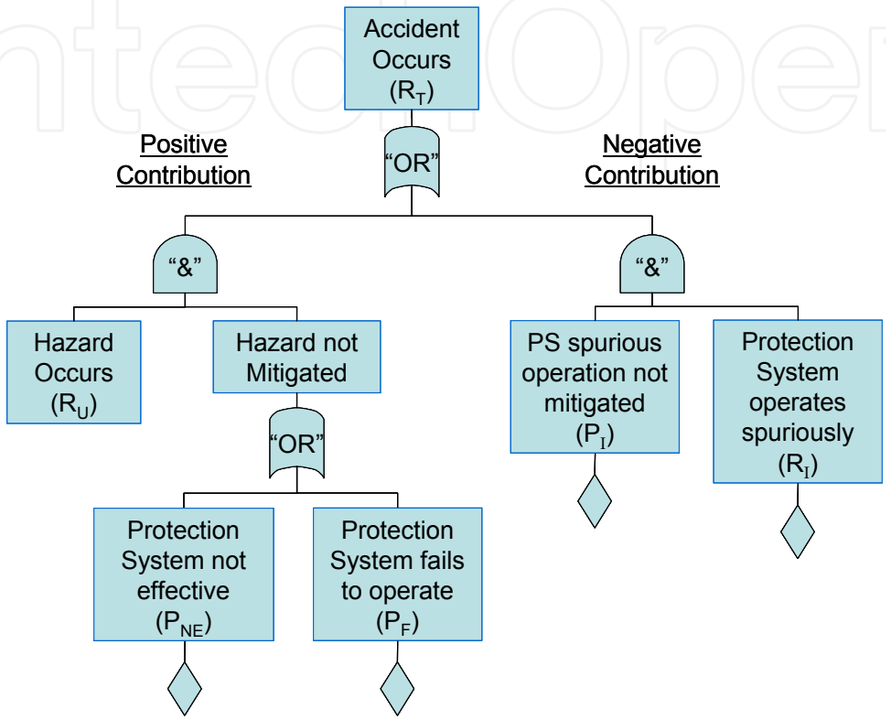

Fig. 4. Simple Accident Fault Tree - Protection System

It is the presence of the external input R_U that distinguishes Figure 4 from Fault Trees in general, and it is this that enables the computation of both the positive and negative contributions to safety.

It should be noted that a simple failure (to operate) of the Protection System is shown on the *success* side of the model rather than on the *failure* side - corresponding to shortening the green arrow on Figure 3 rather than lengthening the red arrow. This would be valid for our airbag if driver behaviour were not affected by the knowledge that the car had an airbag - because the risk of airbag failure would simply be the same as not having an airbag at all for the period of the failure. However, if drivers drove less carefully and / or faster in expectation that the airbag would always protect them in the event of a head-on collision then the consequences (and therefore the risk) from failure of the airbag to operate when required would be correspondingly greater - in this case the effect of the failure would better be shown on the failure side of the model. The latter case is an example of what is very common in safety-related environments, especially where humans are involved, and requires extra care to be taken when incorporating such loss-type failures in a risk model.

In practice, risk models are much more complex than the simple illustration in Figure 4. Their uses range from a discrete safety assessment of part of an overall system up to developing a risk model for an entire operation. An example of the latter use, from the ATM field, is EUROCONTROL's Integrated Risk Picture (IRP) [Perrin & Kirwan, 2009]. As a complete model of both positive and negative contributions to safety, the IRP has proved to be a much more powerful tool, in the management of functional safety, than a simple RCS (see section 3.2.2 above) and can be used in many different ways including *a priori* safety assessment, safety monitoring and safety-strategy formulation. Such models are used also in some parts of the nuclear and rail industries but, to the authors' knowledge, not in other industry sectors at the moment.

4. Requirements engineering – The key to safety assessment

Capturing, and then satisfying, a complete and correct set of safety requirements is as fundamental to any *a priori* safety assessment as requirements engineering is to systems engineering in general, as explained below.

4.1 Requirements capture

Some crucial issues regarding requirements capture can be expressed through the simple, but rigorous, requirements-engineering (RE) model shown in Figure 5. This model has been adapted from [Jackson, 1995], in the introduction to which Dr Jackson sums up the requirements-capture problem perfectly, as follows:

*"We are concerned both with the world, **in which the machine serves a useful purpose**, and with the machine itself. The competing demands and attractions of these two concerns must be appropriately balanced. Failure to balance them harms our work".*

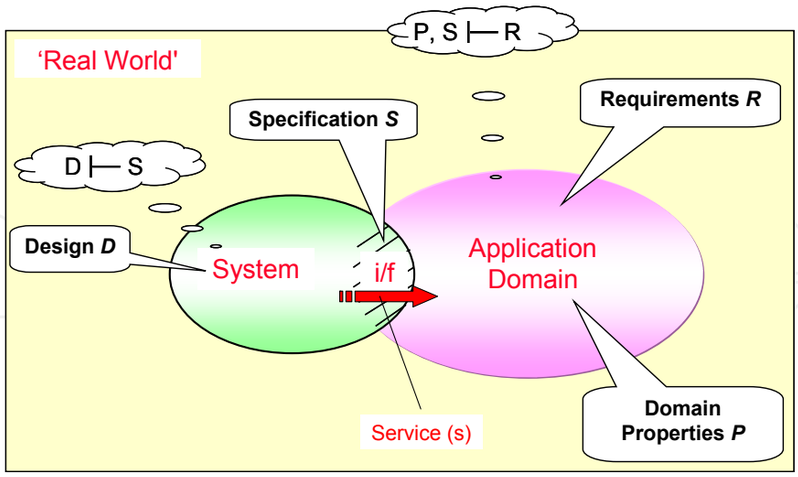

Fig. 5. Jackson Requirements-capture Model - General Form

In this context, what has been said in section 2 above about the lack of a success approach in safety assessments is an example of a pre-occupation with the machine itself at the expense of considering its useful purpose (ie to reduce pre-existing risk). Figure 5 helps clear our thinking as follows.

In the Jackson model, the *system* exists in the *real world*. The part (ie subset) of the real world that influences the system, and into which the system provides a *service* through an interface (*i/f*), is known as the *application domain*. *Requirements* are what we want to make happen in the application domain[12] and are defined in that domain - not in the system.

A *specification* is what the system has to do across the interface in order that the *requirements* can be satisfied - ie a specification takes an external, or "black-box", view of the system. Another way of thinking about a specification is that it contains all the shared properties between the service provider and the service user - therefore it might include things that the service user has to do, not just what the system has to do.

Design, on the other hand, describes what the system itself is actually like and includes all those characteristics that are not directly required by the users but are implicitly necessary in order for the system to fulfil its specification and thereby satisfy the user requirements. Design is essentially an internal, or "white-box", view of the system.

The formal notation in the "bubbles" in Figure 5 defines two relationships that must be shown to be true in requirements capture:

1. that the specification *S* satisfies the requirements *R*. However, this can be true only for a given set of properties *P* of the application domain; therefore, if any one of these three sets of parameters is changed then satisfaction demonstration is invalidated until one of the other sets is also changed
2. that the design *D* satisfies the specification S

The distinction, and relationship, between requirements, specifications, application-domain properties, and design are not merely academic niceties; rather, they provide the essential foundations for developing systems that do, and can be shown to do, everything required.

What is described above in this section applies, of course, to systems engineering in general. However, if the assertion at the beginning of section 1 is correct then it should be possible to apply the same principles to the safety perspective. By comparing Figure 5 with Figure 2 (b) we can see that there is a direct equivalence for safety, as shown in Figure 6.

The main differences from Figure 5 are limited to:

- the Application Domain is now known as the Operational Environment / EUC - merely a change in terminology
- the aspects of the Requirements of particular interest are the Safety Criteria - ie the level of safety that has to be achieved in the Operational Environment / EUC
- for safety-related systems of the type shown in Figure 2 (b) above, the *pre-existing hazards* exist in the Operational Environment / EUC [13] - therefore, the primary Requirements are for the service(s) provided by the system to reduce the associated pre-existing risks to the level defined by the Safety Criteria.

Otherwise, everything that is said in section 4.1.1 above applies to safety.

[12] Since the service users are in the Application Domain these requirements are sometimes called *User* Requirements

[13] Indeed they are the most important properties of the operational environment / EUC!

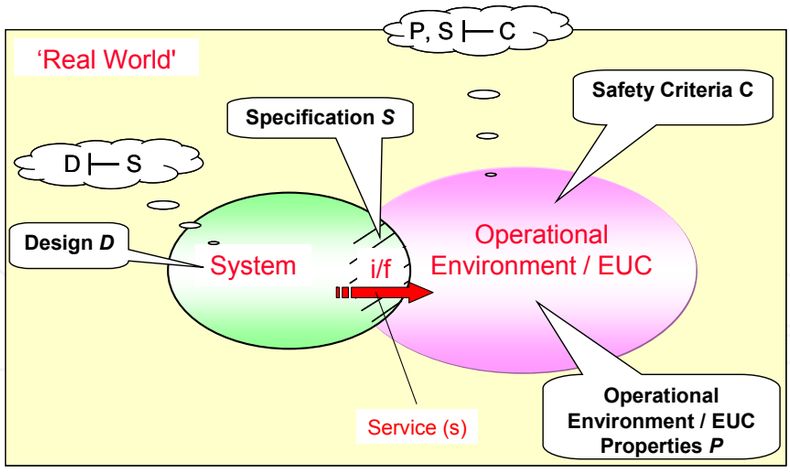

Fig. 6. Safety Engineering form of the Jackson Requirements-capture Model

4.2 Safety requirements satisfaction

Implementation of the design, in the built and integrated system, involves a **third** relationship that must be shown to be true:

1. the implementation *I* satisfies the design *D*

The validity of this relationship requires two objectives to be satisfied in implementation of the design - ie showing that:

- the required properties (functionality, performance, reliability and integrity) of the built system satisfy the requirements established for the design, and
- no emergent properties (eg common-cause failures) and unwanted functionality have been introduced inadvertently such that they could adversely affect the ability of the built system to satisfy the (safety) requirements established for the design.

Because these two objectives are generic - ie apply to all properties of a system - there is no difference in principle between the satisfaction of safety requirements and the satisfaction of design requirements in general. That said, there is usually a difference in degree, in that safety requirements require a higher level of assurance that they have been captured, and then satisfied, completely and correctly.

5. Safety assurance and safety cases

5.1 Safety assurance

Safety assurance, like systems assurance in general, relies on planned, systematic activities to provide the necessary confidence that a service or functional system satisfies its requirements (which are themselves complete and correct), in its intended environment[14]. *Assurance activities* are systematic in that they specify how the *assurance objectives* (ie what has to be demonstrated) are to be achieved, as indicated in Figure 7.

[14] From a safety perspective, this would mean achieving an acceptable or tolerable level of safety - see definition of safety assurance in [European Commission, 2005]

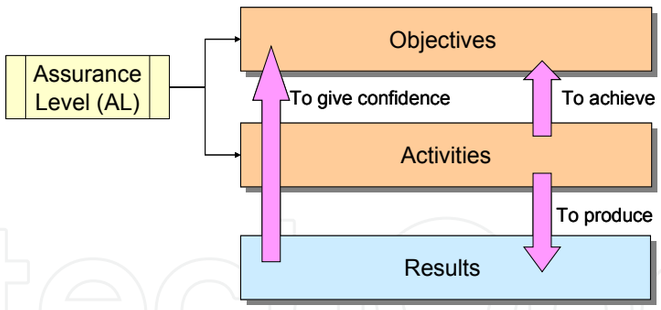

Fig. 7. Safety Assurance – Basic Elements

Which assurance objectives have to be achieved, and the rigour with which they have to be achieved, are often determined by assurance levels (ALs), which are based upon the potential consequences of the anomalous behaviour of the system element concerned, as determined by the system safety assessment process.

The AL implies that the level of effort recommended for showing compliance with safety requirements increases with both the severity of the end effect of the element failure. and the probability / likelihood of occurrence of that end effect, given that the failure has occurred [Mana et al, 2007][15]. The results (outputs) of the activities are then used to show that the assurance objectives have been achieved.

For high-integrity systems in particular, there is a further issue that safety assurance is often used to address and concerns the safety-integrity of system elements - software functions and human tasks, in particular. Whereas it may be necessary to specify Safety Integrity Requirements for all elements of a system in order to show compliance with a numerical Safety Criterion, it is usually very difficult to show in a direct way - through, for example, test results - that such requirements are actually satisfied in implementation. In such situations, it becomes necessary to adopt a more indirect, assurance-based approach which uses the rigour of the development processes to give confidence that the requirements are likely to be / have been satisfied. This is reflected in, for example, airborne software standard DOD 178B [RTCA, 1992] and IEC 61508 both of which are assurance based.

The problem with safety standards is that their use can become highly *proceduralized*, leaving open two important questions:

- where did the full set of assurance objectives come from in the first place?
- how was it decided which objectives and activities have to be done (or may be ignored) for a given AL, and how to we know that this would be appropriate for a particular application?

We can address this problem by putting safety assurance into an *argument* framework but in order to understand this we first need to look have a brief look at Safety Cases.

[15] In some standards, the likelihood of occurrence of the failure is also taken into account - ie the assurance is based on the risk associated with a failure, not just the consequence thereof. This is the case with IEC 61508 and related standards, in which the term SIL (Safety Integrity Level) is used instead of AL.

5.2 Safety cases

Safety assessments are often done within the context of a Safety Case which, like a legal case, comprises two main elements:

- a set of *arguments* - ie statements which claim that something is true (or false), together with
- supporting *evidence* to show that the argument is actually true.

Safety arguments are normally set out hierarchically; this is shown in Figure 8 using and adapted form of goal-structuring notation (GSN). In safety work [Kelly & Weaver, 2004], GSN is simply a graphical representation of an argument / evidence structure and usually starts with the top-level claim (Arg 0) that something is (or will be) acceptably (or tolerably) safe; this is then decomposed such that it is true only if, and only if, the next-level argument statements (in this case Arg 1 to 4) are all true. The *strategy* text should explain the rationale for that decomposition.

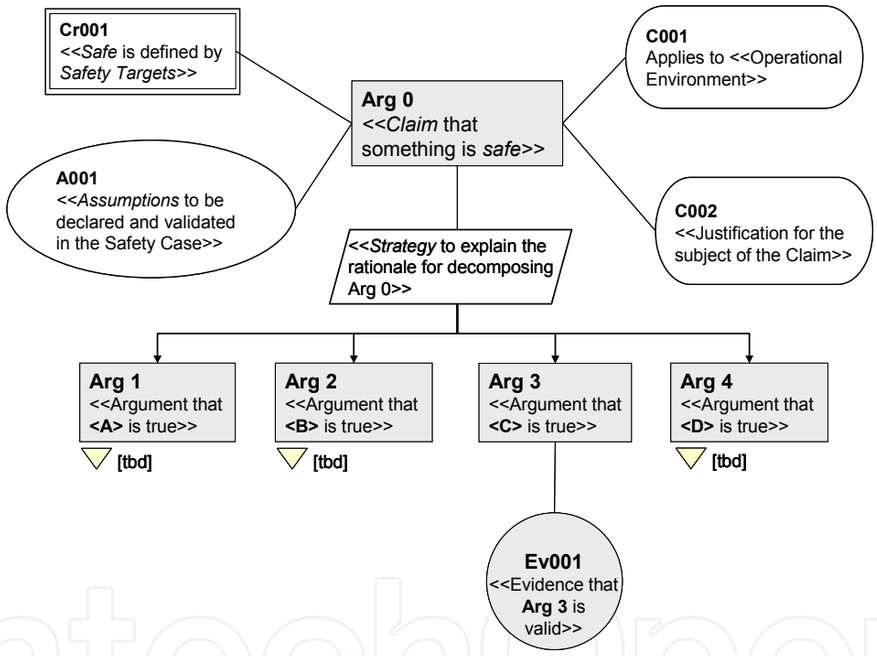

Fig. 8. A generic high-level Safety Argument

The *claim* is supported by vital contextual information, as follows:

- what is meant by *acceptably safe* is defined by means of *safety criteria*
- the *context* for the claim must include a description of the operational environment for which the claim is being made - we can deduce from section 4.1 above how critical this is to the validity of the claim
- *assumptions* are usually facts on which the claim depends and over which the organisation responsible for the safety case has no managerial control - eg driver behaviour, in the case of our airbag
- if the claim relates to a major change to a safety-related system, it is good practice to provide a *justification* for that change.

The arguments would then be further sub-divided until a level is reached at which a piece of documented evidence, of a manageable size, could be produced to show that the corresponding argument statement is valid. The question is how to ensure that a safety argument is complete and rigorous – for this, we use the three formal relationships derived in section 4, as follows.

- **Arg 1** - the system has been **specified** to be safe - ie meets the appropriate safety criteria - in the given operational environment (or EUC)
- **Arg 2** - the system **design** satisfies the specification
- **Arg 3** - the **implementation** satisfies the design

Then, by adding two further arguments:

- **Arg 4 -** the **transition** from current system to the new (or modified) system will be safe – ie the known risks during this process have been reduced ALARP - and
- **Arg 5 -** the **system** will be shown to operate safely **in service**

we have a sufficient, high-level safety argument for developing a new or modified system, bringing it into service and maintaining it throughout its operational life [Fowler et al, 2009]. Since it is the safety argument that determines ultimately what we need to demonstrate, we can use it to drive the whole assurance process as shown in Figure 9.

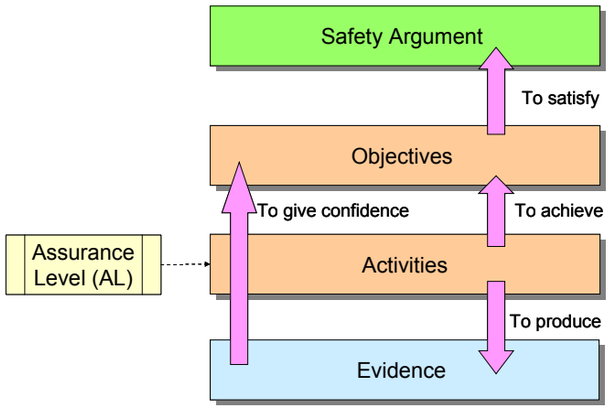

Fig. 9. Safety Assurance within an Argument Framework

The key point about this diagram is that it is the needs of the argument (ie the generation of evidence) that drive the activities - not the other way around - and the lifecycle phases contain only those activities that are necessary to support the argument.

6. Safety assessment in the project lifecycle

The above assurance principles apply to the five phases of a typical project lifecycle, as shown in Figure 10.

In practice, the Safety Plan (produced at the start of a project) should set out a specific safety argument and assurance objectives – with a rationale as to how they were derived to suit the nature and scope of the safety assessment concerned - the lifecycle assurance activities to be carried out, and the tools, techniques etc to be employed. Since the Safety Case (developed during, but finalised at the end, of a project) uses the same argument, it needs only to

present the evidence resulting from the activities and provide the rationale as to how that evidence satisfies the argument.

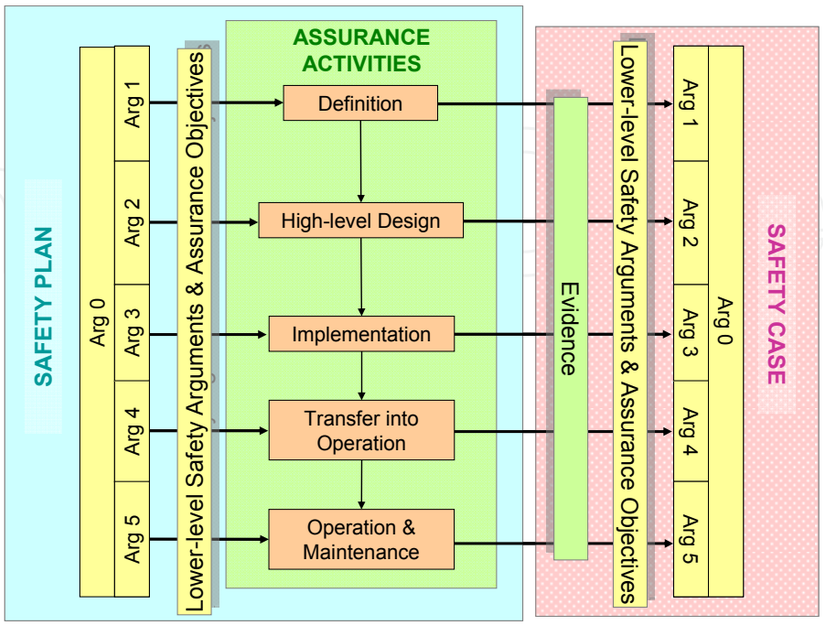

Fig. 10. Overall Safety Lifecycle Process

We will now examine in more detail what is done in the way of safety assurance objectives and activities in each phase of the lifecycle.

6.1 Definition phase

From section 5, we can see that in this phase we need to show that the system has been *specified* to meet the appropriate safety criteria in the given operational environment (or EUC). We will use a new, innovative[16] railway level-crossing[17] control (and possibly protection) system for a planned new two-way suburban road in order to illustrate some of the points in the steps described in sections 6.1.1 to 6.1.4 below - it should be noted that the analysis presented here is not intended to be exhaustive.

6.1.1 Operational environment

For our level-crossing, the properties of the operational environment would need to include:

- users of the crossing - eg passenger and freight trains, road vehicles (of which 80% are cars / light vans and the rest are trucks up to 40 tonnes weight) and occasional pedestrians; exceptionally (once very 1-2 months), large slow-moving vehicles carrying abnormally heavy loads will need to use the crossing
- average length of train - 120 m

[16] This is intended to be a hypothetical example; traditional railway standards - eg [RSSB, 2007] - for level crossings would probably not apply.
[17] Where a railway and road intersect at the same level.

- two-way rail traffic levels - 150 passenger trains per day (mainly between 07:00 and 23:00 hours) and 10 freight trains per day (mainly at night)
- road traffic levels - 3000 vehicles per day (mainly between 07:00 and 23:00 hours)
- traffic performance - the current railway speed limit is 140 kph for passenger trains and 60 kph for freight trains. The planned speed limit for the road is 100 kph for cars, 80 kph for vehicles over 2 tonnes and 65 kph for vehicles over 7.5 tonnes
- details of the road / track layout - the geography of the area makes the construction of a bridge /tunnel prohibitively expensive[18]

6.1.2 Pre-existing hazards

The main pre-existing hazard is " **HAZ_{PE}#1** - any situation in which, on current intentions, a road user and a train would inadvertently occupy the crossing at the same time". The use of "on current intentions" is crucial since it is describing a hazard not an actual accident. We could use mathematical modelling here to estimate the frequency with which this hazard would occur for a completely uncontrolled crossing and hence estimate the pre-existing risk.

6.1.3 Safety criteria

A suitable quantitative criterion would be that the likelihood of an accident involving multiple fatalities shall not exceed one per 100 years, supported by a second, ALARP criterion. However, given a possible range of outcomes of the hazard in this case, it might be appropriate to make use of a suitable RCS (along the lines of Figure 1 in section 3.2.2 above) in order also to set criteria for outcomes of lesser severity[19].

6.1.4 The specification

We recommend the use of the term Safety Objectives to describe the safety aspects of the specification. The reason is that it helps us remember that, in accordance with the Jackson model of section 4.1 above, what we are seeking to do here is describe, from the users' perspective, what the system must do, not to determine how the system will achieve that in its design.

First of all we need to consider the success case and assess how the pre-existing hazard is mitigated for all *normal* conditions in the operational environment - ie all those conditions that our SRS is likely to encounter on a day-to day basis - constructing various operational scenarios (eg single and multiple trains) as necessary. Two examples of a Safety Objective for this are as follows:

- SO#1 - a road user shall not enter the area defined by the crossing until its exit from the crossing is clear
- SO#2 - road users shall not enter the crossing from [say] 1 minute prior to a single[20] approaching train reaching the crossing until the train has cleared the crossing.

[18] This might need to be justified on ALARP grounds.
[19] However, for the purposes of this simple illustration, we will assume that if a train travelling at normal speed collides with a road vehicle there will be some fatalities.
[20] We would need an equivalent rule (ie Safety Objective) for multiple-train situations.

Note that these are genuine objectives (as above) and that the illustrative numbers would need to be verified by some form of dynamic risk modelling based on the environment described in section 6.1.1.

Next we need to assess how well the pre-existing hazard is mitigated for all *abnormal* conditions in the operational environment - ie all those adverse conditions that our SRS might exceptionally encounter - again, constructing various operational scenarios as necessary. An example of a Safety Objective for this is as follows:

- SO#n - in the event that an abnormally large / heavy vehicle (to be defined) is required to use the crossing, all approaching trains shall be prevented from entering the section of track [say] 1 km prior to the crossing until the vehicle has completely cleared the crossing.

This situation is expected to occur infrequently (see section 6.1.1 above) and therefore the system is allowed to operate in a different mode - in this case the train no longer has priority over the road vehicle - from the *normal* case. Again, some form of dynamic risk modelling could be used to determine a suitable exclusion distance for approaching trains.

Finally, we need to consider the potential failure modes of the system, at the service level. At this level, we are not concerned with the causes of failure[21], only with the consequences of failure - for which we would often use Event-tree Analysis for assessing multiple possible outcomes of a particular failure. It is important that the identification of possible failure modes be as exhaustive as possible; a useful starting point is to take each of the success-case Safety Objectives and ask the question what happens if it not satisfied. This will lead to the *system-generated* hazards, an example of which is:

- HAZ_{SG}#1 - road vehicle enters crossing that is in a *closed* state[22] (failure to satisfy SO#2).

Using the operational data from section 6.1.1 we can derive the following Safety Objective to limit the frequency of the hazard such that the appropriate portion of the tolerable-risk criterion (see section 6.1.3) is satisfied for this hazard:

- SO#n+r - the probability of a road vehicle entering the crossing when it is closed shall not exceed $5x10^{-5}$ per operating hour

Note that this illustrative figure takes account of the total number of system-generated hazards (assumed to be four in this illustration), the frequency with which road and rail traffic uses the crossing, and the providential mitigation that even if a vehicle incorrectly enters the crossing there is a significant probability that it would not actually collide with a train. Note also that the hazard occurrence rate is expressed as a frequency even though the SRS is not continuously operating - this was done in accordance with IEC 61508 because the demand rate on the SRS is relatively high (ie up to 150 operations per day).

Thus, at the end of the Definition Phase we should have a set of Safety Objectives which, if they are satisfied in the system design and implementation, would ensure that the pre-existing risk is mitigated, and the system-generated risk is limited, such that the level crossing would satisfy the specified quantitative Safety Criteria.

---

[21] This is done in the failure analysis of the high-level design - see section 6.2 below

[22] *Closed* here is defined by SO#2 (ie from 1 minute before an approaching train reaches the crossing, until the crossing is clear) - it does not necessarily imply a physical closure

6.2 High-level design phase

Having derived what Jackson [Jackson, 1995] refers to as a Specification for the system, the system-development task becomes less safety-specific, and has even more common with general system-engineering principles, except for one key feature - the higher level of confidence (ie assurance) that is required in the results of safety assessment.

Design, as we have seen, is about the internal properties of the system but for this phase we restrict the analysis to a logical design, in which Safety Requirements describe the main human tasks and machine-based functions that constitute the system, and the interactions between them. An illustration, based on our 'futuristic' level-crossing control system (LCCS) of section 6.1, is given in Figure 11.

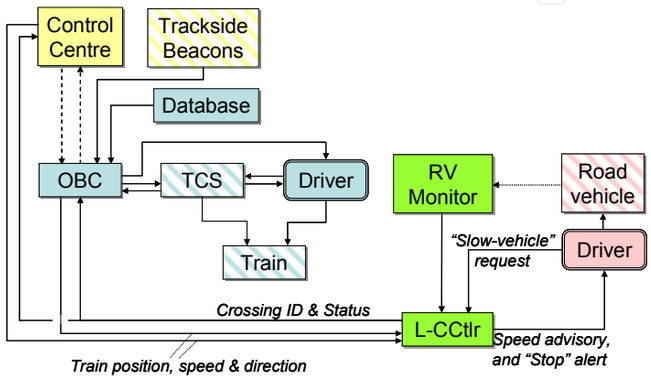

Fig. 11. Example of a Logical Model

A description of this example and the way that the model works is beyond the scope of this chapter - suffice it to say that the new system comprises a fully automated Level-crossing Controller and a Road Vehicle Monitor, which detects the presence of road vehicles within the crossing area. It is to be integrated into a regionally-based "Moving Block" signalling system using Communications Based Train Control - all the train control systems including ATP, but excluding the Onboard Computer, are subsumed into the TCS box[23].

The main points to note are as follows:

- it does not show elements of the physical design[24], such as hardware, software, procedures, training etc - nor does it separately represent human-machine interfaces, these being implicit in every link between a human and machine actor
- since the machine (eg Onboard Computer) and human (eg Driver) elements are shown separately this implies that the degree of automation in the system has been decided, at least provisionally
- the use of colour distinguishes between those elements of the end-to-end system that are in the train (blue), in the road vehicle (pink), in the LCCS (green) and are elements of external, interfacing systems (yellow). All elements that remain unchanged are shown with a striped background.

[23] Although level crossing control would normally be integrated with the Control Centre, for the purposes of this illustration we assume a separate subsystem.
[24] As we will see, physical design is taken to be the first stage of Implementation.

Safety Requirements capture what each of those "actors" needs to provide in terms of functionality, performance, reliability and integrity in order to satisfy the specified Safety Objectives. Whereas Figure 11 shows the actors and the way in which they interact quite clearly, the functionality that they provide is contained in the textural Safety Requirements, and the links between those functions are not easily seen. For this reason, on functionally-rich systems we often use a Functional Model, showing an abstract view of the system functions and data, as a bridge between the Specification and the Logical Design, thus increasing the confidence of the completeness and correctness of the latter [Fowler et al, 2009].

It is very important to note that making an argument for a Logical Design is not simply a matter of showing traceability of the individual Safety Requirements, for the Logical Design, back to the Safety Objectives of the Specification. This would ignore three possibilities: that the design as a whole might be in someway internally incoherent; that new failure properties could emerge at the design level that were not apparent at the higher (service) level; or that the Safety Requirements are too demanding of technology and / or human performance. Thus it is necessary to provide assurance that the Logical Design:

- has all of the functionality and performance attributes that are necessary to satisfy the Safety Objectives of the (service-level) Specification
- will deliver this functionality and performance for all *normal* conditions of the operation environment that it is likely to encounter in day-to-day operations
- is robust against (ie work through), or at least resilient to (ie recover easily from), any *abnormal* conditions that it may exceptionally encounter
- has sufficient reliability and integrity to satisfy the Safety Objectives of the Specification
- is realistic in terms of the feasibility of a potential physical system to satisfy the Safety Requirements, and the ability of validation & verification methods to demonstrate, at the appropriate time and to the necessary level of confidence, that the Safety Requirements are eventually satisfied.

By now it will (we hope!) be no surprise that to analyse, verify and validate the design from a safety perspective we use classical systems-engineering techniques, including:

- requirements traceability [Hull et al, 2005]
- Use-case Analysis [ISO/IEC, 2005] and Fast-time / Real-time simulations - for the normal, abnormal and failure scenarios
- Fault-tree Analysis [IEC, 2006b] to assess the causes of failure, "top down" , and
- Failure Modes Effects & Criticality Analysis [IEC, 2006a] to check the FTA, "bottom up".

Furthermore, since the human elements of the system have started to emerge, we can use human factors (HF) techniques such as Cognitive Task Analysis (CTA) and Human Reliability Assessment (HRA) to assess initially whether the task, performance and reliability & integrity demands which the Safety Requirements place on the human operators are at least realistic.

By the end of the High-level Design Phase we should have a set of Safety Requirements - covering the success and failure cases - that are sufficient to ensure that, if they are satisfied in the Implementation, the specified Safety Objectives would be met.

6.3 Implementation phase

We have defined the Implementation Phase such that it comprises development of a Physical Design and the realisation of the Physical Design in the built system. In making an argument for Implementation, we need to show that:

- the Physical Design satisfies the Safety Requirements for the Logical Design
- the causes or effects of any adverse, emergent safety properties (eg common-cause failures) or unwanted functionality have been mitigated in the Physical Design such that they do not jeopardize the satisfaction of the Safety Requirements
- the built system satisfies the Safety Requirements of the Physical Design (ie verification[25])
- the built and integrated system is consistent with the original qualitative Safety Objectives (ie validation).

In the physical design, we take the Safety Requirements from the Logical Design and allocate them to the elements of the Physical System, as follows:

- human tasks map on to, and provide the initial Safety Requirements for, skills, knowledge, procedures and training
- machine-based functions map on to, and provide the initial Safety Requirements for, hardware and software design
- human-machine interactions map on to, and provide the initial Safety Requirements for, human-machine interface (HMI) design.

These in turn lead to further design, Safety Requirements derivation and implementation for each of these elements and then to integration of the complete system - for further reading see [ISO/IEC, 2008b and ISO/IEC, 2008a]. The steps would follow, for example, the classical "V-model" of system development in which the safety engineer must ensure that the physical system as a whole (and its constituent parts) have sufficient reliability and integrity, and complete and correct functionality and performance, to satisfy the higher-level Safety Requirements. These are discussed in turn, as follows.

6.3.1 Building reliable systems – General

The engineering of a safety related system must ensure, as a minimum, that the safety reliability and integrity requirements are met. It is useful to consider first how failures occur and what techniques can be used to reduce failure rates to meet the safety criteria.

Random failures can occur in hardware of any kind due to physical degradation and wear-out mechanisms; the exact time when such a failure will occur is unknown but statistical distributions can be used to predict failure rates and quantification of overall system reliability can be modelled by techniques such as FTA mentioned earlier. Techniques for making hardware elements sufficiently reliable are considered in section 6.3.2 below.

Systematic failures by contrast are caused by design defects – they will occur whenever a system enters a state in which a latent defect is revealed. Software failures are always

[25] It is acknowledged that, in some industries / countries, verification and validation may have the opposite meanings to those used herein.

systematic[26], but hardware designs (especially those such as computer processor chips) can also exhibit systematic failures. Methods of ensuring that software is sufficiently free of defects, such that it can meet its safety function and performance requirements with sufficient reliability and integrity, are discussed in section 6.3.4 below. The concept of a systematic failure may also be applicable to human factors - eg if a procedure is designed incorrectly.

Common-cause failures are ones in which redundant subsystems fail at the same time due to the same external events (eg earthquake or tsunami), internal causes (eg power supply failure) or due to a systematic error affecting multiple systems (known as a common mode failure). This could be a software defect or a human maintenance intervention, for example. Common cause failures in practice often limit the achievable reliability of complex systems. The general approach is to attempt to identify and eliminate sources of common cause failure where possible and also to be conservative with reliability predictions to cater for the possibility of unknown causes.

6.3.2 Hardware safety

The main techniques for ensuring that random hardware failures are sufficiently unlikely are use of high reliability components and redundancy. High-reliability components are expensive so redundancy is used in practice except in special circumstances (such as space applications) where repair is difficult or impossible. Redundancy simply means having two or more subsystems each of which can perform the required safety functions; if one fails then a standby can take over provided that there is some mechanism (automated or manual) to detect the failure. Further gains in reliability can sometimes be achieved by using diversity, where the standby system(s) are not identical to each other. Ideally the diversity should be both conceptual (using different physical processes or measurements) and methodological (different design methods) since this helps to reduce the likelihood of common mode failures (discussed in the next section). Part 2 of IEC 61508 [IEC, 2010], in particular, provides requirements and guidance on hardware safety techniques and measures.

Of course, we must not forget the fundamental principle that (with the possible exception of Information Systems) the functionality and performance properties of hardware is as important to safety as its reliability and integrity is - see the airbag example in section 2.

6.3.3 Human factors safety

HF is a topic that in the past in many industries has had only scant coverage in functional safety [Sandom, 2002]. More recently, things have improved, not the least in European ATM for which EUROCONTROL has developed the "HF Case" [EUROCONTROL, 2007].

In the HF Case, Human Factors are considered on two different levels, the System Level and the Human Performance Level. The underlying philosophy is that the design of tasks, procedures and tools must correspond to the safety requirements on both the level of the overall system as well as on the level of the individual human operator.

[26] Although they may be revealed in a quasi-random way.

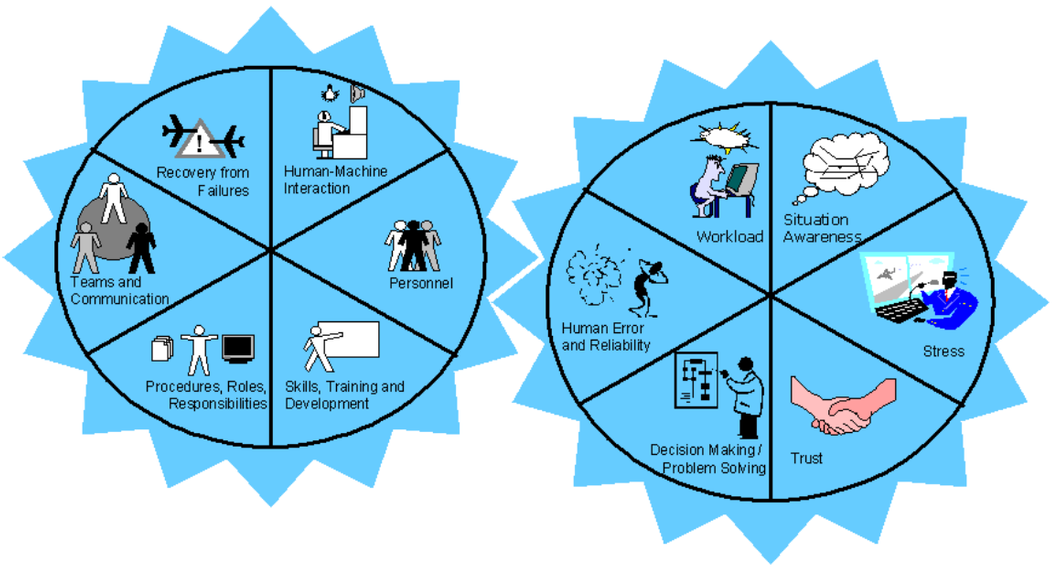

Fig. 12. The HF "Gearbox"

The HF aspects can be classified into six areas at each level, and for each specific task, procedure or tool some of twelve areas may be more important than others; however, the underlying principle is that the HF evaluation should involve both levels. The resulting approach is known as the HF Gearbox, as illustrated in Figure 12.

From a safety perspective, the HF Gearbox, at the human-performance level, addresses:

- *Situation awareness*: the continuous extraction of external information, integration of this information with previous knowledge to form a coherent mental picture, and use of that picture in directing further perception and anticipating future events
- *Workload*: the degree to which the operators' mental resources are consumed by the task at hand.
- *Stress*: a positive or (more often) negative emotional response to a more or less demanding situation.
- *Trust*: the 'right degree of trust' in the equipment permits the operator to exploit the benefits of a equipment whilst being aware of its limitations
- *Human error and reliability*: Human reliability is the operator's capacity to execute a specific task resulting in a performance within specified limits - see section 6.2.
- *Decision making/problem solving*: Decision making is performing a selection between possible options in a specific situation. It can be rational / or emotional.

From a safety perspective, the HF Gearbox, at the system level, addresses:

- *Human-Machine Interaction*: input devices, visual displays, information requirements, alarm handling, HMI usability, fatigue, distraction and concentration, noise, lighting and general comfort as it affects human performance
- *Organization and Staffing*: Staff requirements, manpower availability, operator profile / selection criteria, shift organization
- *Training and Development:* Training needs performance / competence standards, training content, training methods and media, trainer role / responsibilities /

competency, on-the-job training, emergency / abnormal-situation training, testing of training effectiveness, effects on operational task performance
- *Procedures, Roles and Responsibilities*: allocation of task, involvement, workload, trust / confidence, skill degradation, procedure format and positioning, procedure structure, procedure content, procedure realism
- *Teams and Communication*: Team structures / dynamics / relations, coordination, handover processes, communication workload, phraseology, language differences, communication methods, interference effects, information content
- *Recovery from Failures*: Human error potential, error prevention / detection / recovery, detection of, and recovery from, equipment failures.

It is crucial from a safety perspective that HF is not considered to be a separate activity - rather it must be fully integrated into the safety assessment and resulting safety case.

6.3.4 Software safety

6.3.4.1 An IEC 61508 perspective on software safety

Part 3 of IEC 61508 [IEC, 2010] starts at the point where the software requirements specification for a safety related system has been developed, in terms of safety functional behaviour and safety integrity. The system engineering approach described in this chapter should, of course, be used in deriving those software requirements in the first place.

IEC 61508 Part 3 is based on a conventional V lifecycle model for the design and implementation of software. A process-based approach is convenient for software developers who have to follow the standard, but Part 3 stresses that it is not the existence of the process itself but the evidence resulting from the *application* of the process which will demonstrate the achievement of safe software. The development lifecycle stages comprise:

- software architectural design,
- detailed design,
- module design,
- coding.

Verification is required after each stage, to check that the output of the design stage is a correct refinement of the input and has other necessary properties. Verification as a minimum includes software code reviews but can include other forms of analysis ranging from code complexity analysis and rigorous inspection techniques up to formal proof that the software has certain properties. Testing of the software mirrors the design, as follows:

- module or unit testing, to demonstrate that the each software module behaves in accordance with its specification
- integration test at the software design level(s) (which includes hardware/software integration testing), to show that all modules function together as intended
- safety validation testing at the software requirements level, to provide confidence that the safety function and performance requirements are met.

The stress laid by IEC 61508 Part 3 on module testing is justified by experience that software which has been well tested at the module level will usually reveal few errors during later testing, although it is a step often skimped due to time pressures during development.

Detailed techniques and measures are recommended in Annexes A and B of IEC 61508 Part 3 to support each lifecycle stage. Techniques are either Highly Recommended (HR), Recommended (R) or noted without any specific force (-). In a very small number of cases, techniques are Not Recommended. The nature and rigour of the HR techniques and the number to be applied increases with the SIL – this is common to a number of software safety standards and guidelines. However, it is not possible simply to apply all the HR techniques for a given SIL, since some may be contradictory; therefore judgement must be applied in selecting the techniques which will give the greatest benefit. If a relevant HR technique is not used, its omission must be agreed with the independent safety assessor (see below). The standard stresses that it is the combination of testing and analysis that provides the necessary confidence that the software will be safe.

Properties of the design artefact(s) are stated for each lifecycle stage as a guide to selecting and justifying the techniques and measures to be used. Properties include:

- completeness (with respect to the higher level representation)
- correctness
- ease of understanding
- freedom from intrinsic errors.

An "intrinsic" error is one which can be recognised regardless of the functions which the software is to realise – examples at the source code level could include division by zero, numeric overflow, or access via a null pointer. This class of errors typically cause run-time "crashes" and there are analytical tools which can help to eliminate such errors.

Use of pre-existing software elements (such as an operating system or communications software) is allowed, provided that sufficient evidence of reliable operation can be provided. This can include evidence from non-safety applications subject to certain conditions.

Although IEC 61508 Part 3 is based on the V lifecycle, alternative lifecycles can be used. For example, if code is generated automatically from a high-level requirements model (which is possible for some types of control applications) then the software design and coding stages can be omitted, although testing will still be required. The selection of tools to support each lifecycle stage and technique is specifically addressed in Part 3 – a tool which makes a direct contribution to the final software (such as a compiler) must be chosen and justified with greater care than one where the output of the tool is readily amenable to manual inspection and correction.

In common with Parts 1 and 2 of IEC 61508, independent functional safety assessment is required, the degree of independence depending on the SIL to be achieved. For example, a person independent of the designer is sufficient at SIL 1 whereas an independent organisation is required at SIL 4. The assessor should examine both the process itself (the selection of techniques and measures) and the products of the development (for example the test and analysis results and the backing evidence that the testing and analysis have been sufficiently thorough). It is interesting to note that independence between designer/implementer and verifier/tester is not required in IEC 61508, although it is required in other standards and guidelines for safe software.

All evidence resulting from test, analysis and field service experience must be recorded and kept under configuration management along with the design artefacts and software code–

therefore, applying the standard should generate the evidence required to meet SW01 (see subsection 6.3.4.2).

6.3.4.2 A regulatory perspective on software safety

There are two main ways of gaining **assurance** that the safety requirements have been properly and completely implemented in an SRS: assurance which is obtained directly from the attributes of the product itself; and that which is obtained from the characteristics of the processes which gave rise to the product.

So what is an appropriate balance between product and process assurance, and what part should the various standards play in the achievement and demonstration of system safety? The SW01 section of the UK CAA's safety regulation CAP 670 [UK CAA] takes an objective-based approach that provides a sensible answer to these questions[27]. SW01 takes an approach to safety assurance which is deliberately non-prescriptive in terms of the development process; instead, it demands arguments and evidence of the achievement of five safety assurance objectives – ie to show that the:

- software safety requirements correctly state what is necessary and sufficient to achieve tolerable safety, in the system context
- the software satisfies its safety requirements
- each software safety requirement can be traced to the same level of design at which its satisfaction is demonstrated
- software implemented as a result of software safety requirements is not interfered with by other software [that is not safety related]
- all assurance relates to a known executable version of the software, a known range of configuration data and a known set of software products, data and descriptions that have been used in the production of that version.

SW01 defines seven behavioural *attributes* of safety-related software, which the safety assurance must address, with equal rigour (or a valid argument presented as to why a particular attribute has not been addressed), as follows: functionality, timing, accuracy, robustness, reliability, resource usage, and overload tolerance.

In the context of requirements satisfaction, SW01 allows assurance to be offered from three different sources – ie testing, analysis (of design), and field service experience (FSE). For each source of assurance, two forms of evidence are required, for each *attribute*, as follows:

- *Direct* evidence - that which provides actual measures of the product (software) attribute concerned and is the most direct and tangible way of showing that a particular assurance objective has been achieved
- *Backing* evidence would relate to quality of the process by which those measures were obtained and provides information about the quality of the *direct* evidence, particularly the amount of confidence that can be placed in it.

Testing is restricted largely to tests of the final product (executable code) or a very close relation of it. *Direct* evidence is concerned with what tests were carried out and what the

[27] Although SW01 covers specifically safety-related software, most of the principles in it can apply equally to the wider aspects of the system

results showed in terms of satisfaction of the safety requirements. *Backing* evidence is concerned with showing that the tests were specified correctly and carried out adequately.

FSE is based on previous operational use of the software. *Direct* evidence is concerned with analysis of data from FSE and what the results of that analysis showed in terms of satisfaction of the safety requirements. *Backing* evidence is concerned with showing that the environment from which the data was obtained is sufficiently similar to that to which the re-used software will be subjected, that an adequate fault-recording process was in place when the software was originally deployed, and that the data-analysis process was adequate and properly carried out.

Analysis covers any proof of requirements satisfaction that is obtained from the design or other representation of the software, including models, prototypes, source code etc. *Direct* evidence is concerned with what the results of the particular *analysis* techniques showed in terms of satisfaction of the safety requirements. *Backing* evidence is concerned with showing that design and other representations of the software were appropriate and adequate and that the *analysis* was adequately specified and properly conducted; where *analysis* was carried out on source code, it is necessary also to show that the object code correctly translates the source code.

In general, the rigour demanded of the evidence increases as the integrity required of the software increases. However, SW01 defined this integrity only in terms of the consequence of failure – ie it does not take account of the probability that such a failure will occur, as is the case, for example, in IEC 61508.

The way in which evidence from the three sources can be used in combination varies according to the attribute for which evidence is offered, and also depends on the integrity required of the software. As the required integrity increases, SW01 allows less dependence on a single source of evidence, and places more emphasis on *analysis* of design as the primary source of evidence.

The advantage of testing over design analysis is that it is carried out on the end product rather than on a representation of that product. On the other hand, the effectiveness of testing can be limited by problems with test coverage and with confidence levels in respect of statistical attributes of the system. For these reasons, assurance from testing usually takes second place to design analysis for the more safety-critical systems, though it is interesting to note that SW01, for example, mandates some degree of testing even where the primary source of assurance is design analysis.

As a source of assurance, design analysis has one further advantage over testing – it is available much earlier in the lifecycle and can therefore make a major contribution to the reduction of programme risk. Iterative development techniques seek to bring forward the availability of test evidence but in so doing bring with them their own problems, including a possible reduction in effectiveness of design assurance unless specific measures are taken to avoid this.

6.4 Transfer phase

The Transfer-into-Operation Phase takes the assurance process up to the point that the system concerned is ready to be brought into operational service. In making an argument for Transfer into Operation, we need to show that:

- everything necessary has been to prepare the new (or modified) system for operational service
- the process of bringing the system into service – ie transitioning from the current system to the full new system – will itself be as safe as reasonably practicable

Preparation for operational service is about showing that all the necessary operational procedures, trained personnel and technical resources are in place to operate and maintain the system and that, under conditions as close a possible to real operational service, the system / resources as a whole will meet the expectations of the service users (ie system validation). Typically, a safety case would be submitted for management approval and regulatory endorsement before transition from the current system to the new system begins. Therefore it is necessary to show beforehand that any risks associated with transition will be managed such that the AFARP criterion will be met throughout this process.

Especially in safety-critical industries that require an uninterrupted operation for 365 days per year, live system testing and the subsequent transition from the old to the new system are often hazardous. Thus a Transition Plan needs to drawn up and executed to ensure that:

- the hazards and risks associated with the transition process have been completely identified and correctly assessed
- measures have been put in place to reduce those risks *ALARP*
- contingency plans are in place to revert to a safe state if the transition is not entirely successful
- the old system components can be removed safely

6.5 Operational phase

The safety focus during the whole of the systems in-service life is on providing assurance of its continuing safe operation. This is vital for a number of reasons, including:

- the *a priori* safety assessment, covered in Arg 1 to 3, might not be complete and correct in every particular
- the system, including the equipment and human elements, might degrade in operational service
- the system will most probably be subject to changes at various times during its operational life
- the operational environment might change during the life of the system.

Thus we need to ensure first of all that the system (comprising equipment, people and procedures) will be supported so as to maintain the required level of safety. The evidence in support of this will be mainly in the form of SMS processes (and related operational and engineering procedures) and how it will be ensured that they will be properly applied - including the use of surveys and audits related to the application of those SMS processes.

Secondly, in order to provide assurance of actual safety achievement we need to show that:

- there is a culture to encourage full and accurate reporting of safety incidents
- the frequency of safety incidents will be measured against pre-defined indicators
- all reported incidents will be properly investigated
- appropriate corrective action will be taken to prevent incident recurrence.

Finally, we need to show that there are procedures in place to manage future changes to the system and / or it operational environment.

7. Conclusions

We started by asserting that safety is not a separate attribute of a system but is a property that depends on other attributes, and on the context in which the system is used. The misconception that adequate reliability and integrity are sufficient to ensure the safety of a system has been prevalent in, for example, the ATM sector [Fowler & Grand-Perret, 2007], but is dispelled in the specific context of software by the results of extensive research [Leveson, 2001] and more generally herein by rationale argument using the example of a car airbag.

After introducing some basics safety concepts and principles, we then showed that safety is as much dependent on correct functionality & performance of the system as it is on system reliability & integrity - the former set of attributes being necessary for the mitigation of *pre-existing* risk (inherent in the operational environment) and the latter for controlling *system-generated* risk (caused by system failure). This led to the view that what was needed was a broader approach (what we called the *success & failure* approach) to system safety assessment, a view that was then shown to be consistent with the principles underlying the generic functional-safety standard IEC 61508 [IEC, 2010] - principles that, however, are not always captured in industry-specific instantiations of this standard.

We then turned to a vital aspect of systems engineering - ie requirements engineering, some important principles of which are advocated in [Jackson, 1995] - and found direct equivalence between the derivation of the required safety properties of a system and the derivation of its non-safety properties.

Finally, we introduced the principles of safety assurance and safety cases and showed how they should drive all the processes of a safety assessment, throughout the lifecycle. Whilst emphasising the importance of ensuring that the level of assurance is appropriate to the safety-criticality of the system, we now leave it to the knowledgeable systems engineer to recognise the common processes of a system development lifecycle in this chapter and to conclude for himself / herself that safety is (with the assurance proviso) actually just a viewpoint (albeit a very important one) on systems engineering!

8. Acknowledgment

The authors would like to record their appreciation for the many very helpful suggestions made by Drs Michael A Jackson and Anthony Hall during the development of this chapter.

9. References

European Commission, 2005, Regulation (EC) No 2096/2005, *Common Requirements for the Provision of Air Navigation Services*, published in the Official Journal of the European Union

Eurocontrol, 2007, *The Human Factors Case: Guidance for Human Factors Integration*

Eurocontrol, 2010, http://www.skybrary.aero/index.php/Risk_Assessment

Fowler D, Grand-Perret S, 2007, *Penetrating the Fog of Safety Assessment - and Vice-versa*, Proceedings of the 2nd IET International Conference on System Safety, London, UK

Fowler D, Perrin E, and Pierce R, 2009, *2020 Foresight - a Systems-engineering Approach to Assessing the Safety of the SESAR Operational Concept*, Proceedings of the 8th USA/Europe Air Traffic Management Research and Development Seminar, Napa, USA

Hull E, Jackson K and Dick J, 2005, *Requirements Engineering*, Springer, ISBN 1852338792

International Electrotechnical Commission, 2006a, IEC 60812, *Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA)*

International Electrotechnical Commission, 2006b, IEC 61025, *Fault Tree Analysis*

International Electrotechnical Commission, 2010, IEC 61508, *Functional Safety of Electrical/ Electronic/ Programmable Electronic Safety Related Systems*, V 2.0

ISO/IEC 12207, 2008a, *Systems and Software Engineering - Software Lifecycle Processes*, V2.0

ISO/IEC 15288, 2008b, *Systems and Software Engineering - System Lifecycle Processes*, V 2.0

ISO/IEC 19501, 2005, *Information technology, Open Distributed Processing – Unified Modelling Language (UML)*, V 1.4.2

Jackson M A, 1995, *The World and the Machine*, Proceedings of 17th International Conference on Software Engineering, IEEE, pp283-292

Kelly T and Weaver R, 2004, *The Goal Structuring Notation – a Safety Argument Notation*, http://www-users.cs.york.ac.uk/~tpk/dsn2004.pdf

Ladkin P B, 2008, *An Overview of IEC 61508 on E/E/PE Functional Safety*, http://www.causalis.com/IEC61508FunctionalSafety.pdf

Leveson N G, 2001, *The Role of Software in Recent Aerospace Accidents*, 19th International System Safety Conference, Huntsville AL, USA

Mana P, De Rede J-M and Fowler D, 2007 *Assurance Levels for ATM system elements: Human, Operational Procedure, and Software*, proceedings of the 2nd IET International Conference on System Safety, London, UK

Perrin E and Kirwan B, 2009, *Predicting the Future: The Integrated Risk Picture*, Proceedings of the 4th IET International Conference on System Safety Engineering, London, UK

Pierce, R and Fowler D, 2010, *Applying IEC 61508 to Air Traffic Management*, Proceedings of the Eighteenth Safety Critical Systems Symposium, Bristol, UK

Rail Safety and Standards Board - RSSB (2007). *Engineering Safety Management (The Yellow Book), Volumes 1 and 2 - Fundamentals and Guidance*, Issue 4

Reason J, 2000, http://www.bmj.com/cgi/content/full/320/7237/768

RTCA, 1992, DO-178B, *Software Considerations in Airborne Systems and Equipment Certification*,

Sandom C, 2002, *Human Factors Considerations for System Safety*, in Components of System Safety, Redmill F and Anderson T [Eds.], proceedings of 10th Safety Critical Systems Symposium, 5th-7th February 2002 Southampton, Springer-Verlag, UK

UK CAA, 2011, UK Civil Aviation Authority, CAP670, *Air Traffic Services Safety Requirements*

Systems Engineering - Practice and Theory
Edited by Prof. Boris Cogan

ISBN 978-953-51-0322-6
Hard cover, 354 pages
Publisher InTech
Published online 16, March, 2012
Published in print edition March, 2012

The book "Systems Engineering: Practice and Theory" is a collection of articles written by developers and researches from all around the globe. Mostly they present methodologies for separate Systems Engineering processes; others consider issues of adjacent knowledge areas and sub-areas that significantly contribute to systems development, operation, and maintenance. Case studies include aircraft, spacecrafts, and space systems development, post-analysis of data collected during operation of large systems etc. Important issues related to "bottlenecks" of Systems Engineering, such as complexity, reliability, and safety of different kinds of systems, creation, operation and maintenance of services, system-human communication, and management tasks done during system projects are addressed in the collection. This book is for people who are interested in the modern state of the Systems Engineering knowledge area and for systems engineers involved in different activities of the area. Some articles may be a valuable source for university lecturers and students; most of case studies can be directly used in Systems Engineering courses as illustrative materials.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen