Challenging Safety Regulation – a Wake-up Call

Derek Fowler, Carl Sandom and Alan Simpson, Praxis Critical Systems Ltd, Bath, UK

Keywords: safety, assurance, regulation, ATM

Abstract

New objective-based safety regulatory requirements for European air traffic services mandate the introduction of safety management systems, set a numerical target level of safety (TLS) for service provision, and specify requirements for the approval of software in safety-related systems.

These requirements challenge much current "best practice" in safety management and, though directed at air traffic management (ATM), have major implications for other application sectors.

In presenting a framework for tackling these issues, the paper examines current safety practices. In particular it challenges, from a theoretical perspective, and by reference to recent research, two common misconceptions - that safety is largely a matter of equipment reliability and that process-based assurance can provide adequate evidence of system safety. It argues that failure to address these two weaknesses can lead to, wasted effort, inadequate safety assurance and, at worst, unsafe systems.

Since sound safety requirements are fundamental to system safety, the paper proposes a rigorous requirements-engineering method and explains how *satisfaction arguments* can help ensure correctness, completeness, and consistency throughout high-level safety requirements, system specification and system design, in relation to the important (often overlooked) properties of the application domain.

The paper asserts that it is all too common, in the ATM sector at least, for more than 90% of system safety effort to be expended on less than 10% of the problem (ie on equipment issues) and argues that a service-level TLS requires that the dominant cause of accidents (human factors) be given proportionate consideration, otherwise the overall system may not achieve the required level of safety.

Objective-based safety regulation and effective safety management systems place the "burden of proof" in safety assurance firmly on the "regulatee", and demands a highly rigorous approach to system development and safety assurance. The paper outlines the principles of "human-centred engineering" and "white-box safety" and explains how they can make a major contribution to the achievement and assurance of safety.

The paper concludes by considering whether the new regulatory regime is *a good thing*.

Introduction

Safety regulation is the oversight of the safety of an organization whose facilities or services could impact significantly on the safety of the environment in which they operate.

Traditionally, in many industries including ATM, safety regulation was done prescriptively – ie the regulator defined the rules and standards to be followed, used audit and inspection to check compliance with them, and quite commonly would issue a safety certificate to that effect. In so doing, the regulator implicitly (if not explicitly) inherited a substantial part of the responsibly from the regulatee. That required a great deal of specialist resource on the part of the regulator and was often over-constraining for the regulatee, particularly in the introduction of new processes and technologies.

In Europe ATM, for example, recognition of these difficulties has led to a recent trend towards objective-based safety regulation in which safety is much more clearly the responsibility of the ATM service provider, the regulator's role being mainly to ensure that the service provider discharges his responsibilities properly. The regulator sets objectives for the achievement and demonstration of safety and the service provider has to show (by argument and evidence) that he has met those objectives - the *burden of proof* rests entirely with the service provider. The use of standards may still be appropriate but the service provider has to show that

the standards he chooses to use are appropriate – not merely claim compliance with them.
In a world of objective-based safety regulation, appropriate top-level safety objectives for demonstrating that a system was tolerably safe would be to show that both of the following are true:

- a complete and correct set of safety requirements exists, sufficient to enable the required level of safety;
- those safety requirements have been met in the implemented system.

Although that thesis appears to be self-evident, the practical realisation of those two fundamental assurance principles can vary enormously within and across various industries. There is a popular belief that safety is largely a matter of reliability and that safety can be delivered merely by adherence to prescribed processes, especially in relation to software development. As will be explained, theory and experience have already shown this to be a very narrow view of safety - the introduction into European ATM of a numerical TLS and objective-based software safety regulation will prove it to be seriously flawed.

Current Safety Assurance Practice– the Pitfalls

Reference 1 – "Logic versus Magic in Critical Systems" - points out the dangers of being seduced by complex highly technology and placing faith in 'magic' tools and development processes for assurance of safety, instead of using logical reasoning as to why a system can be considered safe. Some current manifestations of the magic problem are as follows.

Safety is Reliability: In her paper on aerospace software safety (ref 2), Nancy Leveson presents compelling evidence, based on her review of major software-related accidents, that software reliability has **never** been the cause of such disasters. On the contrary, the software has in every case performed in exactly the manner that it was designed to – the problem was not that the software was unreliable but that the software was inadvertently designed to do the wrong thing for the circumstances in which it "failed". Why then, she asks, is so much faith placed in process-based safety standards, adherence to which would merely ensure that such software would do the wrong thing more reliably?!

IEC 61508 (ref 3), for example, states that the primary purpose of a safety-related system (SRS), and/or the components within it, is to reduce risk, and therefore that safety requirements must be specified as: the *safety functions* that are needed to in order to provide risk reduction; and the *integrity* required of those safety functions. In other words, it recognises that the safety of a system depends as much on **what** is does as on how **reliably** it does it. Unfortunately, the Standard then goes on to relate *necessary risk reduction* to Safety Integrity Levels (SILs) and to define SILs in terms of probability of failure – ie reliability – and most of the rest of the Standard is about specifying processes for SRS development, according to the required SIL – largely ignoring the functional aspects of safety.

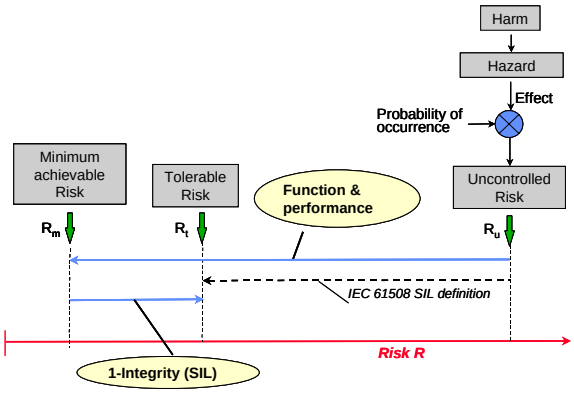

Figure 1 – Functionality and Integrity

Reference 4 presents an alternative model, as shown in Figure 1. It argued that it is specified functionality and performance of an SRS that determines the maximum risk reduction that can be achieved, and that SRS failures erode that potential reduction. Therefore, if the SRS functionality is insufficient to achieve the *necessary risk reduction* then, no matter how reliable an SRS was, it would not be safe – a conclusion that is entirely consistent with Nancy Leveson's findings.

IEC 61508 is not alone in this – DO178B (ref 5) and other commonly used safety standards suffer from the same process-biased view of safety. Compliance with these standards, whilst giving some assurance of SRS integrity, does little to assure the correct functionality and performance of the SRS and therefore that they are really safe. Yet claims such as "Product X meets IEC 61508 SIL3" are not uncommon – "so what?" would be an appropriate **logical** response to such claims!

Technology is the Answer to Safety: Most analyses would lead to the conclusion that ATM, as a safety function, should be of high integrity – say, IEC 61508 SIL 3. Yet it is not uncommon for ATM equipment to be based on millions of lines of software, written in weakly typed, non deterministic languages such as C or C++, in highly distributed architectures. Furthermore, in the interest of increasing traffic capacity, and based on evidence (eg ref 6) that (fallible) human operators are responsible for well over 90% of aircraft safety incidents, more and more automation is being built into ATM equipments. Of course, it would be impossible to reason logically that such complex systems are safe and therefore developers resort to the magic of CASE tools, code generators, automated testing and compliance with process-based standards as the basis of safety arguments. Where such arguments are found wanting, it is not uncommon to resort to the claim that the (now infallible!) human operator will intervene in the event of equipment failure – this, despite the fact that the very automation whose failure the human is trying to mitigate has raised traffic levels, and de-skilled the operator, so much that such intervention is becoming increasingly difficult.

Safety Requirements Derivation Made Easy: *Risk classification* schemes appear to provide an easy way of converting the assessed severity of a hazard into a tolerable frequency of occurrence and hey presto to a safety integrity requirement. However, as reference 7 points out, such schemes present many a trap for the unwary and before using them it would be pertinent to ask, at least:

- where the conversion values used in the scheme came from and whether they are (still) valid;
- at what level in the system hierarchy the values apply;
- to what operational environment the values apply – eg type of airspace, traffic patterns, traffic density, spatial dimension, phase of flight etc;
- how total risk can be deduced from analysis of isolated hazards, in fragments of the total system.

The imminent introduction of numerical TLS for European ATM means that time is running out for the magicians! The TLS specifies a maximum probability of collision of 1.55E-8 per flight hour, and being set at the service level, is defined outside of the boundary of the entire ATM system. Therefore, since the primary purpose of ATM is to reduce risk to below the TLS, all those properties of all elements of the ATM system that affect the achieved risk reduction now have to be considered, in an integrated manner. The task soon facing ATM service providers will be to:

- set risk targets for an ATM system, that are, sufficient, achievable, and verifiable against the TLS
- demonstrate that those risk targets have been met in the complete system.

The paper now considers how these challenges will require a more logical approach to the specification and realisation of safety requirements and to the assurance of system safety.

Safety Requirements Engineering

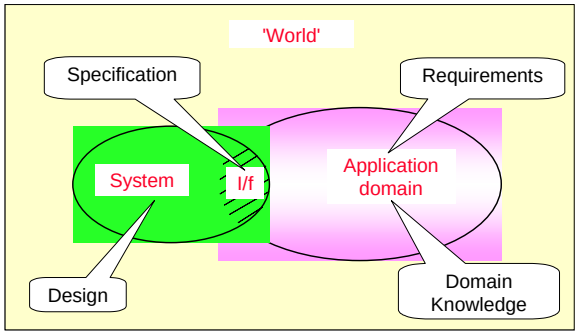

Figure 2 – Basic RE Model

Figure 2 shows a simple, but rigorous RE model, based on Praxis' Reveal® approach. Systems exist in the *world*, and in general consist of people (operators) and procedures as well as equipment. That part of the *world* that influences, and/or is influenced by, the *system* is known as the *application domain*. Users and other stakeholders exist in the *application domain*. The *system* interacts with the *application domain* through an *interface* (I/f). *Requirements* are what we want to make happen in the *application domain* – ie *requirements* are defined in the *application domain*, not in the *system*. This view is entirely consistent with risk-reducing view of an SRS given in IEC 61508.

A *specification* is what the System has to do across the *interface* in order that the *requirements* can be satisfied – ie *specifications* are defined at the *system* boundary and take a "black-box" view of the system.

Most importantly, a *specification* is a necessary but not sufficient condition for *requirements* satisfaction. It is also necessary to state the assumptions and other pre-defined information - called *domain knowledge* - about the *application domain* on which the relationship between *requirements* and *specification* depends – eg the structure of, and rules applicable to users of, the airspace. *Domain knowledge* is often taken for granted and sometimes overlooked completely; however, without it, the *satisfaction argument* – that the *requirements* are satisfied by the *specification* - is not valid. Ariane 5 is a good example of where failure to recognise *domain* changes (from Ariane 4) led disaster (ref 2).

Design describes what the *system* is actually like and includes all those characteristics that are not explicitly required by the *AD* but are implicitly necessary in order for the *system* to fulfil its *specification* and thereby help satisfy the *requirements*. Design is essentially an internal, or "white-box", view of the system

The distinction and relationship between *requirements*, *specifications*, *domain knowledge* and *design* are not merely academic niceties but provide the essential RE foundations for developing systems that do, and can be shown to do, everything required of them (see ref 8). This is now examined in the context of safety.

Accidents, Hazards and Causes: An *accident* is an unintended event that results in death or serious injury. *Accidents* occur in the real *world*. A *hazard* is a system state that can lead to an accident, and has the important characteristic that it is described only at the boundary of the *system*, as indicated in Figure 3.

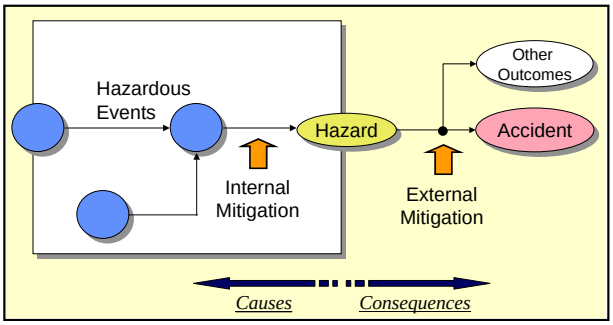

Figure 3 – Hazard Model

Once a *hazard* has occurred, the *system* has no control over the consequences – ie it has in itself no means of stopping an *accident* occurring, although external mitigation (including pure chance) may be possible. Failures within a system that may cause *hazards* are called *hazardous events* and it is important to distinguish them from *hazards*. *Hazardous events* are properties of the *design* of the system and they determine the likelihood of occurrence of a *hazard*. It is important to note also that hazard analysis of the *system* itself cannot give a full picture of what the *system* is required to do, merely how reliably it must do it. Therefore, hazard analysis must start in the *application domain* in order to determine the primary, functional safety requirements for an SRS.

Safety Requirements Determination:

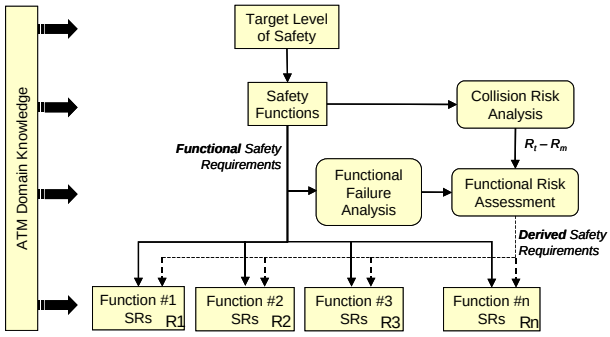

Figure 4 – Safety Requirements Determination

Figure 4 illustrates the process of getting from a TLS, to a set of system safety requirements, developed for a recent ATM project (ref 9). The first step is to identify the *safety functions* that will reduce risk in the *application domain*, and then to carry out some form of collision-risk analysis (CRA) to show that the functional and performance properties of those *safety functions* are sufficient to reduce the risk below the level specified by the TLS; those properties form the *functional safety requirements* for the respective system functions. A Functional Failure Analysis (FFA) provides a *severity* level for each potential mode of failure of each system function. The CRA also needs to indicate by how

much the risk is reduced below the TLS (see $R_t - R_m$ on Figure 1 above) before the *safety integrity requirements* (ie the allowable rate of occurrence of each function failure mode) can be determined, taking account of:

- the severity of the failure concerned
- any additional *safety functions* available to mitigate the consequences of failure
- the integrity required of the overall system – ie $R_t - R_m$

The *safety integrity requirements* and the properties of the additional *safety functions* are known collectively as *derived safety requirements*; together with the initial *functional safety requirements*, they form the full set of safety requirements (R1, R2 ..Rn).

The final step in the safety requirements process is to show, via a *satisfaction argument*, that the system safety requirements are sufficient to meet the TLS, as illustrated in Figure5.

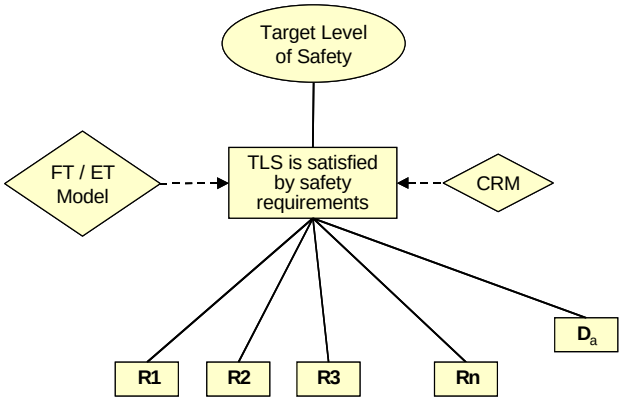

Figure 5 – Satisfaction Argument

The *satisfaction argument* will be that the TLS is met by the safety requirements (R1 to Rn) given the domain knowledge (D). In this example, the *satisfaction argument* is supported by the results of a (validated) collision-risk model (CRM) to show that the *functional safety requirements* are sufficient, and a Hazard / Risk model (using, say, Fault Tree and Event Tree analyses) to show that the *safety integrity requirements* are also sufficient. D includes issues such as the type, structure and rules of the airspace concerned, the prevailing traffic conditions, and the applicable separation minima; clearly it must also be complete and correct otherwise the *satisfaction argument* is made invalid.

Safety Requirements Realisation

The first stage in the realisation of safety requirements (SRs) is to carry out an architectural design and produce SRs - or, more correctly, safety specifications (SSs) - for the resulting subsystems, as in Figure 6

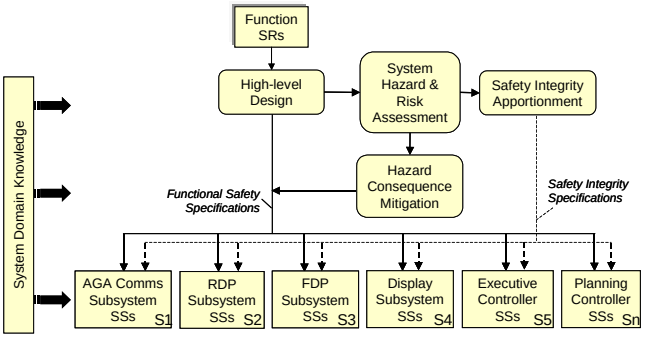

Figure 6 – System Architectural Design

The process is similar in principle to that for safety functions, described above. Firstly, the *functional SRs* for each *safety function* are allocated to the subsystem(s) on which they are to be implemented. The severity of the hazards of failure of each subsystem (defined at the subsystem boundary) are then assessed, any mitigations (of the consequence of failure) are identified and allocated (as additional *functional SRs*), and the *safety integrity requirements* for each subsystem determined. It is important to note that only the *functional SRs* are passed from system to subsystem level; the subsystem *safety integrity requirements* are not inherited from the system level but rather are *derived*, together with the additional, hazard-mitigating, subsystem *functional SRs*, from the hazard and risk assessment. The SS for each subsystem must therefore specify:

- the safety functions to be performed by each subsystem, and the performance (accuracy, response, time, throughput etc) required of them such that the system-level *functional SRs* are met

- the integrity required of each function such that the system-level *safety integrity requirements* are met
- the interactions and interfaces between the subsystems, to implement the system-level *safety functions*

Those three perspectives must be addressed for each subsystem irrespective of whether it is hardware-based, computer-based or human-based, as discussed further below.

Finally, a *satisfaction argument* is required, in order to provide assurance that each system-level SR is met by the subsystem safety specifications (S1 to Sn) given the system domain knowledge (D1 to Dn), as shown in Figure 7. Two types of model to support the satisfaction arguments are suggested in the illustration – one to demonstrate the functional correctness of the SSs and one to demonstrate that the specified subsystem safety integrity and hazard mitigations fully satisfy the system-level *safety integrity requirements*. Again, the adequacy of the domain knowledge is vital. It would also be sensible to consider at this stage whether the resulting SSs are sensible and appropriate according to, for example, equipment technology and human capabilities. This is especially important for aspects such as human performance and reliability where demonstrating compliance is dependent largely on empirical evidence. As indicated by the 'greyed' areas of Figure 7, it should be possible (indeed is essential!) to show complete and continuous satisfaction from the TLS to the set of subsystem SSs.

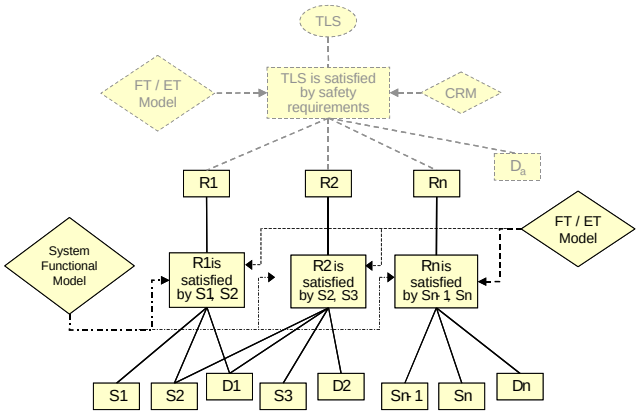

Figure 7 – Satisfaction Argument

Having produced proven SSs for the each subsystem, it is now appropriate to consider how those SSs are realised in the subsystem design. In principle, the process comprises further iterations of the system design and assurance process outlined above, for each subsystem and for the interfaces between them and to the outside world. The particular aspects for equipment and human subsystems are discussed as follows.

Equipment Subsystems: Technological changes in the last 10 years have seen a vast increase in the complexity of systems and the desire to use commercial off-the-shelf (COTS) components, and for systems to implement functions of varying safety criticality on a common platform.

What most safety standards define is a list of techniques that may or may not be applicable to the processes of specification, design, development, verification and validation of the system. Whereas these techniques represent good practice for the particular implementation technology, they focus on the application of the process providing some indication of a more reliable product but falling well short of a logically reasoned argument as to why a system satisfies all of the required safety attributes. Furthermore, one of the major problems with process-based assurance is that any component with an uncertain pedigree (e.g. COTS) is very unlikely to have evidence available to demonstrate that it meets the prescribed process criteria.

The recently issued UK ATM software regulatory requirements (SW01) in Reference 10, require a more logically reasoned approach to safety than required by these processed-based standards ; in outline:

- evidence has to be presented in two forms: *direct* evidence, being the most direct, product-based way of showing that a particular objective has been achieved; and *backing* evidence, providing additional, process-based information, about the trustworthiness of the *direct* evidence.
- all the software's function, performance and integrity safety properties have to be addressed.
- for high-criticality software more weight has to be given to assurance from design analysis than from testing or previous usage.

At the moment only software is covered but system-level and hardware equivalents of SW01 are planned. Figure 8 shows how the rigorous approach outlined above can be applied to a typical equipment subsystem. An architectural design of the subsystem is produced and the functional aspects of the SSs are allocated to the major elements of the subsystem design – ie the hardware and software components, and the interfaces between them. A hazard and risk analysis of potential failures within the subsystem is then carried out to determine the severity of the failure concerned; any additional *safety functions* and/or design techniques available to mitigate the consequences of failure are identified **before** the integrity required of each element necessary to satisfy the subsystem-level *safety integrity requirements* is determined. This is part of the *White Box Safety (WBS)* approach to safety, developed originally as part of research conducted by the UK Defence Evaluation & Research Agency and Praxis Critical Systems into the certification of Integrated Modular Avionics architectures (ref 11). The basic tenet of WBS is to seek ways to exploit the inherent relationship between the safety properties required of a system and its design, its architecture and the processes used to develop it, thereby providing a more robust, well-reasoned argument for the safety of the system, and possibly a reduction in the cost of development and safety approval. The term *White Box Safety* was coined in recognition of the major contribution that design and design assurance (by definition *white-box* activities) can make to the achievement and assurance of safety. The *WBS* approach is based on:

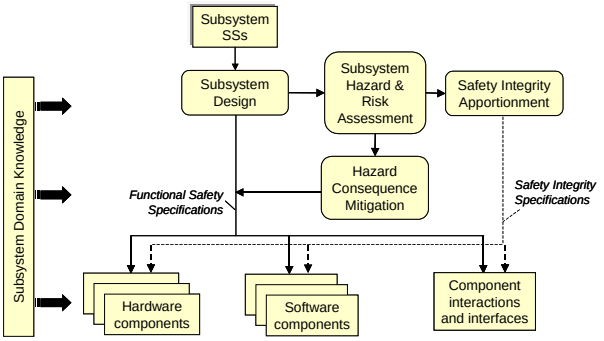

Figure 8 – Equipment - based Subsystem Design

- a rigorous approach to the specification of SRs and SSs, taking full account of the properties of the system and subsystem application domains, as outlined above.
- the optimum use of design techniques (eg redundancy, partitioning, dissimilar software etc) in order to reduce the consequence of failure and of rigorous design methods, component selection, and development processes to reduce the probability of failure; however, it is important to be able to argue independence - ie that there are no common errors, or dependencies between errors – and use this information to, for example, determine the most effective architecture, define the scope, type and purpose of development process activities, and to specify tests.
- experience (ref 11), which has shown that sensible design can not only produce high-integrity systems from largely lower-integrity components (including software languages, COTS and items of unknown pedigree) but also demonstrate the achievement of these high integrity levels with a higher degree of confidence than is possible with components which require high individual integrity.
- the axiom that, since the intrinsic safety properties of a system depend on design at least as much as on anything else, then proving that the design solution satisfies its specification must be a valuable (if not the **most** valuable) source of safety assurance; design analysis techniques vary according to the attribute concerned and include formal proof, simulation, modelling, mathematical proof, prototyping, hardware reliability analysis, software code analysis etc.
- the view that testing (ie validation that the implementation of a design meets its specification and ultimately the original overall requirements), being essentially a *black-box* activity, is rarely (if ever) sufficient to assure the safety of a complex, high-integrity systems – especially those including software and human elements – because of, for example, practical limits on test duration for random properties, and concerns about test coverage for systematic failures; testing is most useful in confirming the results of design analysis (especially for those properties which cannot be verify fully at the design stage) and to ensure that the design has been implemented correctly.

The *WBS* approach is entirely consistent with the principles of SW01, outlined above. Experience of using it has also shown that the cost of development and certification activities can be reduced, whilst still producing a robust, well-reasoned argument for the safety of the system.

Human Subsystems: At its simplest form, the model for the development of human-centred subsystems is similar to those already discussed, as shown in Figure 9. However, that is where the similarity ends! While hardware (and to a lesser extent software) engineering is relatively mature and well understood, that is not the case with human factors. It is generally very difficult to predict all the possible mental states of a human operator in a complex system; even if it were not so, the unpredictability of human behaviour remains. The requirements placed on a human-based subsystem (as set out in the SS) comprise typically:

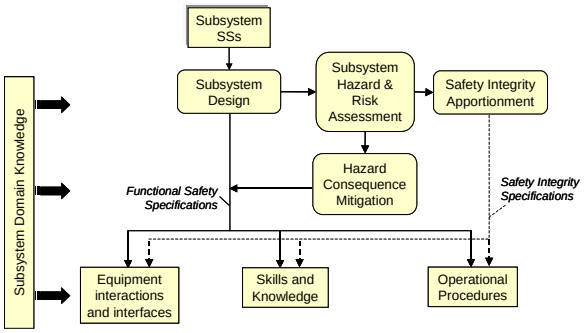

Figure 9 – Human- based Subsystem Design

- the performance of specific functions, directly (eg short-term conflict detection) or indirectly through the operation of equipment; and/or
- the mitigation of the consequences of failures that occur in other subsystems.

In allocating the functional aspects of the SSs it to the elements of the subsystem, and carrying out a hazard and risk assessment, two particular problems needs to be addressed: the complexity of the interactions between the human operator and the equipment subsystems, and the complex nature of human failure. It was to address these problems that Praxis' developed *CONTEXT*® - a systematic and pragmatic framework for the elicitation, specification and management of Human Factors in high-integrity systems, two aspects of which are discussed as follows.

Interactions: *CONTEXT*® uses, inter alia, the concept of *Situational Awareness (SA)* (ref 12) to model human-machine interactions, based on a 3-stage iterative process in which the human operator **samples** the external world situation, **updates** his awareness of that situation and then **acts** to change the situation in order to achieve a particular objective – eg the command and control of aircraft as illustrated in Figure 10.

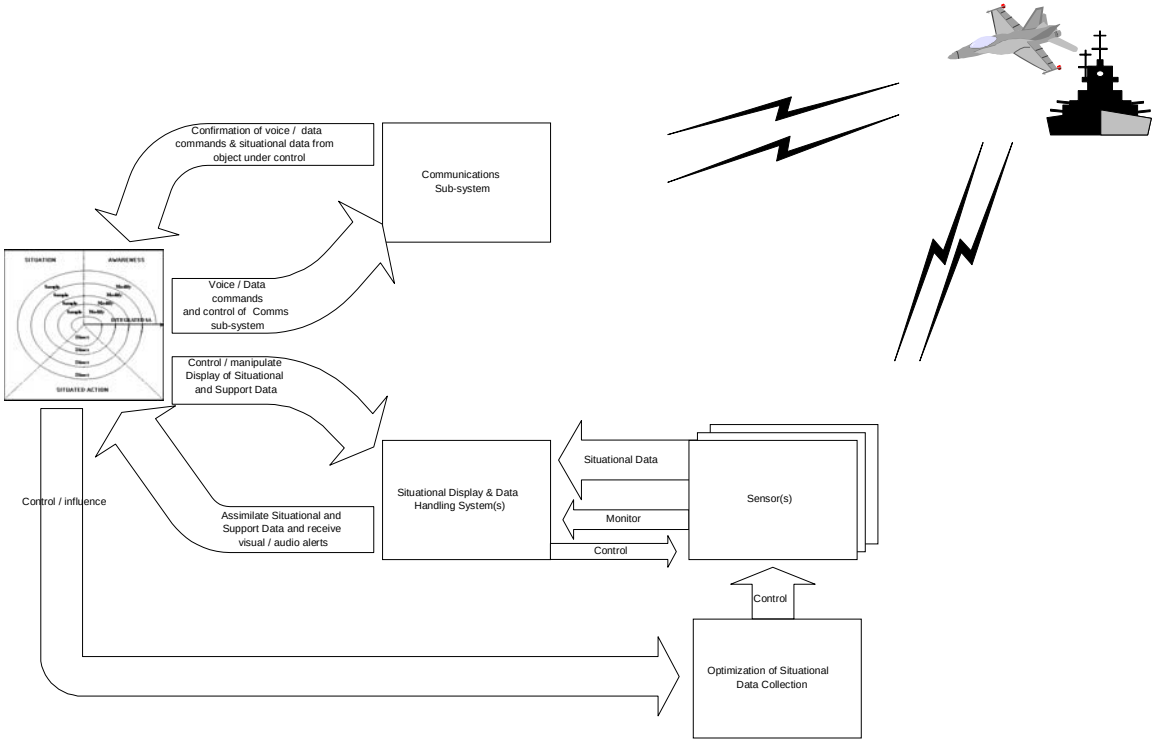

Figure 10 – Illustrative Situational Awareness Model

SA and other systematic approaches to Human Factors in complex systems helps to address issues such as:

- which functionality can reasonably be left to the skills and knowledge of the operator, which additionally has to be written in procedures, and which is best allocated (in part) to other subsystems
- which information the human-computer interface (HCI) must present to the operator, and in what form
- what level of performance can reasonably be expected of the operator, using the combination of HCI, skills, knowledge, and procedures – eg the reasonable maximum number of aircraft that a Controller can handle at one time, for a sustained period, taking account of the prevailing complexity of traffic flows
- what additional tools etc need to be provided in order to improve the performance of the human operator

Human Error Modelling: Much of the literature on human error does not make it clear that people make errors for varying reasons and that the action required to prevent or reduce occurrence, or mitigate the consequences, of one type of error may not be the best way of dealing with other types. On the contrary, it is common for system hazard analysis to stop once a critical human action is identified. More usefully, Kletz (ref 13) characterises human error as:

- *mistakes* - those that occur because someone does not know what to do – ie the intention is wrong.
- *non-compliances* - those that occur because someone knows what to do but decides not to do it, often because the person genuinely believes that departure from the rules, or usual practice, is justified.
- *mismatches* - those that occur because the task is beyond the ability of the person asked to do it.
- *slips* - or momentary lapses of attention. – ie intention is correct but it is not carried out.

A complete understanding of the ways in which people fail, and the most effective means of reducing the consequences sand/or probability of the various types of failure is clearly fundamental to the achievement of the safety in human-based systems. Evidence to underpin a human factors argument should ideally include both formative (generated during the overall system design process) and summative (generated during the evaluation of the final product).

The targeted application of Human Reliability Assessment techniques (see Reference 14 for further description), supported by HCI prototyping and simulation of the full set of human-machine interactions, can provide a useful insight into possible design changes to, for example, human-computer interfaces in order to reduce the probability of human failure, as well as providing 'order of magnitude' probabilities of human failure, which can be compared with the safety integrity targets for the specific subsystem function. However, that is only part of the problem and the contextual nature of human factors means that more formal consideration needs to be given also to organisational and cultural issues in order to demonstrate that the environment in which humans perform safety-related tasks is deliberately designed and maintained in a way that will help reduce human error. The assessment of organisations against a *Safety Maturity Model* (equivalent to the existing SEI Software Capability Maturity Model) would be a useful way of providing (albeit qualitative) evidence of organisational safety.

Conclusions

The introduction of a numerical TLS, backed by the requirement for formal safety management systems and objective-based software safety approval regulations will force ATM service providers to take a more holistic and approach to safety management based on rigorous, logical reasoning as to why systems are safe.

In particular, current practices of considering the human operator as a user rather than as an integral part of the system, viewing safety as largely a matter of reliability, and reliance on largely process-based safety standards, will fall well short of what is needed to meet these new regulatory challenges.

There will be (indeed has been!), some resistance to these initiatives. Unfortunately, expressing the TLS to two decimal places credits it with more precision than it merits, and there are issues in the details of the UK's SW01 software safety approval regulations which leave them vulnerable to some criticism. However, it is the authors' view that the actual value of the TLS and the details of SW01 are largely irrelevant –

rather it is the existence of a service-level safety target, the principles behind SW01 and the requirement for formal safety management, addressed in a logical and rigorous framework as outlined in this paper, which will lead to a more focused, complete and uniform approach to assuring the safety of the travelling public.

References

1. Peter Amey, Logic Versus Magic in Critical Systems. Proceedings of the 6th Ada-Europe International Conference, Leuven, Belgium May 01.
2. Nancy G Leveson, The Role of Software in Recent Aerospace Accidents, Proceedings of the 19th International System Safety Conference, Huntsville, Alabama, USA Sep 01
3. International Electrotechnical Commission, IEC 61508, Functional Safety of Electrical/ Electronic/ Programmable Electronic Safety Related Systems, 65A/254/FDIS, IEC:1999.
4. Fowler D, Application of IEC 61508 to Air Traffic Management and Similar, Complex, Critical Systems – Proceedings of the 8th Safety-Critical Systems Symposium, UK, Feb 00.
5. DO-178B/ ED-12B, Software Considerations in Airborne Systems and Equipment Certification, 1992.
6. CAA, 1998, Analysis of Airprox (P) in the UK: Join Airprox Working Group Report No. 3/97, September 1997.
7. Fowler D, Tiemeyer B, Eaton A, Safety Assurance of Air Traffic Management and Similarly Complex Systems, Proceedings of the 19th International System Safety Conference, Huntsville, USA Sep 01
8. A J Simpson and J. Stoker, Will it be Safe? An approach to Engineering Safety Requirements, Safety Critical Systems Club Symposium, February 2002.
9. Fowler D, and Tiemeyer B, Reduced Vertical Separation in European Airspace – A Case Study in Effective Safety Management, Proceedings of the 20th International System Safety Conference, Denver, USA Aug 02.
10. UK Civil Aviation Authority, CAP 670, Air Traffic Services Safety Requirements, Amdt 3, Sep 99
11. M Ainsworth and A J Simpson, Integrated Modular Avionics — A View on Safe Partitioning, in Towards System Safety, ed. F Redmill and T Anderson, Springer-Verlag, 1999.
12. Sandom C, Situational Awareness Through The Interface: Evaluating Safety In Safety-Critical Control Systems, IEE Proceedings of: People In Control - An International Conference On Human Interfaces In Control Rooms, Cockpits And Command Centres, University Of Bath, UK, June 1999.
13. T. A. Kletz, An Engineer's View of Human Error, Institution of Chemical Engineers, Rugby, UK, 2nd edition, 1991.
14. Kirwan B: A Guide to Practical Human Reliability Assessment, Taylor and Francis, London 1994.

Biographies

The authors are all consultant engineers with Praxis Critical Systems Ltd, 20 Manvers Street, Bath, BA1 1PX, UK, telephone - +44 1225 466991, facsimile - +44 1225 469006

Derek Fowler has over 30 years experience in aerospace and defence, as a chartered systems engineer and project manager. Since 1990 he has specialised mainly in ATM systems, for the last 4 years acting as a safety consultant in that field. e-mail - derek.fowler@praxis-cs.co.uk

Carl Sandom has considerable experience of safety-related development projects. He now specialises in the fields of Safety and Human Factors, a subject area in which he has carried out research and has been awarded a PhD. e-mail: carl.sandom@praxis-cs.co.uk

Alan Simpson is an experienced chartered safety engineer with 15 years experience in critical systems engineering and project management. He has detailed knowledge of aircraft, air traffic management, automotive and railway systems. e-mail: alan.simpson@praxis-cs.co.uk