Human Factors and Software Factors – A Powerfully Reliable System

Richard Maguire B.Eng M.Sc C.Eng MIMechE
SE Validation Limited
Salisbury
United Kingdom
rlm@sevalidation.com

ABSTRACT

There is a great deal of apprehension in engineering circles when new safety systems are being developed which need to have reliance on both humans and software. Many standards and guidance papers in industry provide a great deal of advice on avoiding hazards, problems and mis-haps. Little guidance is given on promoting the economic benefits and synergies of having these two groups of sub-systems working together. Each sub-system has its own performance niche and reliability properties, which can be exploited for the benefit of the whole system. This exploitation will be developed by assessing sample requirements placed on a safety system, and will give guidance on how the requirements may be distributed over the human sub-system and the software sub-system.

A powerful safety and reliability argument will be constructed demonstrating how human factors and software factors can be combined together, utilising the performance capabilities of each. Several examples are given from across industry, with the safety and reliability arguments demonstrated using logical operators. This is then interrogated to demonstrate the positive economic nature of cost, time and quality properties. The results give key guidance on the limits and power of the constructed safety and reliability arguments.

1. INTRODUCTION – A BRIEF HISTORY OF RELIABILITY DEVELOPMENT

Reliability as a technical concept appeared shortly after World War One as a discussion on the reliability comparison of single, twin and four engine aircraft. During World War Two the engineering of the 'V' series missiles led to the development of the product probability law of series component by Robert Lusser [Høyland & Rausand 1994]. During the 50s and 60s the space race was the focus for further reliability research and the 70s and 80s continued with nuclear power and oil and gas engineering. The 90s have seen the fields of human factors and software factors take their turns in the reliability stakes. The current generic definition for reliability given in BS 4778 gives the position as;

Reliability: The ability of an item to perform a required function, under given environmental and operational conditions and for a stated period of time.

The definition need not be uniquely specific to an individual item – i.e. a sub-system, or whole system may also have properties of reliability.

As can be seen over the decades, reliability is at the core of every major industry in the world today – that is why it is so vital to economic prosperity of the first world and the progressing economic development of the third world.

2. THE PROBLEMS OF RELIANCE ON HUMANS AND SOFTWARE

The effectiveness of many human-with-software systems depends critically on how well the needs, capabilities and limitations are taken into account during the design of systems and their related operation and maintenance. Systems that appear to be effective under normal conditions may exhibit significant shortcomings under abnormal or emergency use. Often, system failures may be attributed to 'human error' or similar concepts, however it is infrequent that user error is proven to be the sole cause of these problems [MoD 2004].

The consequence of system failure can be costly in terms of penalties, legislative censure or financial and human losses – a life simply cannot be recovered in the next fiscal quarter. Many system failures could have been prevented or the consequences reduced if greater time and attention had been paid to human factors [MoD 2004] and software factors.

There are several areas where humans are better than computers and vice-versa, software gives repeatable and fast function; humans are better at learning and anticipation. Task allocation needs to be understood and implemented with great care. This is especially true for systems with multi-year acquisition processes where the user and designer are unlikely to share the same temporal space of a project lifecycle.

3. GENERIC DESCRIPTION OF HUMAN-SOFTWARE SYSTEM FAILURE MECHANISM

In a system composed of human and software components it is perfectly fair to state that a failure mechanism may be initiated by either. Software does not usually break or wear out, it fails by design – i.e. software does exactly what it is ordered to do even if it is wrong. This is potentially another form of human error, but at the coding stage of the project, very much earlier than operator errors. Complex software will have multiple inter-relationships, so many that all of them simply cannot be interrogated for reliable performance. There will always be areas of code that remain unchecked. However, it should be noted that software errors do not cause accidents, it is software failures that do that.

Software can also carry out vital performance, safety and reliability checks. Time-stamping messages to allow delay checking; check-sums to allow corruption/quality checking and data format barriers to allow syntax type and size checking. Many of these techniques are applied to the poor old human operator, there is also the case that the software should perform the checks on itself – providing common cause failures can be addressed.

The human-in-the-loop part of the system is unfortunately cited for the majority of unintentional harm in our Armed Forces [Brain & Maguire 2005]. Slips, trips and lapses can all add up to a pretty unreliable performance, especially when considered alone. It is the checks by other parts of the system (other humans as well as software) that prevent all human frailties from propagating to full-blown disasters.

However, it must also be stated that human vigilance is probably responsible for preventing far more harm than all the equipment-based safety features combined [Brain & Maguire 2005]. Situational awareness and 'over-hearing' have been demonstrated to be very real safety and reliability factors [Wright et al 1999].

The dual system reliability roles played by humans and software can be represented graphically using influence diagrams such as Fault-trees. Using Fault-tree influence diagrams does give the opportunity to investigate the relationship between the four components of system reliability discussed above: human error, software error, human vigilance and software vigilance. Figures 1 and 2 show two possible relationships of the four factors, the main difference is in the logical

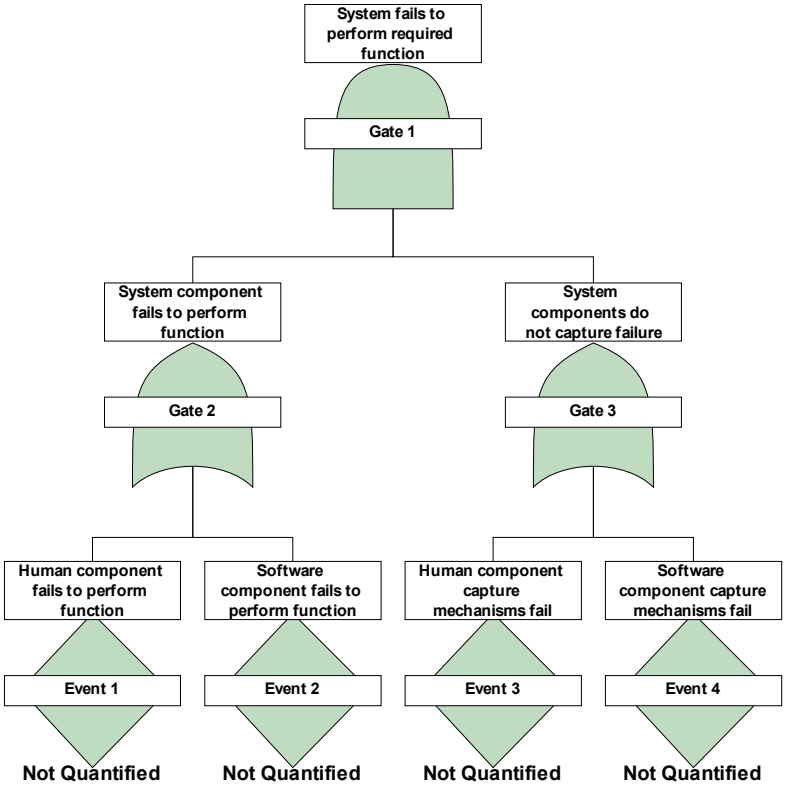

Figure 1 : First arrangement of system component capture mechanisms – 'OR' gate

organisation of the operators in the diagram. Both figures show that failures can originate from either the human components OR the software components of the system. Figure 1 shows that the failure of capture mechanisms, vigilance and performance checks can, in a similar way, originate from either the human OR software components. This is consistent with task allocation practices where barriers to failure and unreliability may be shared between sub-systems of the whole. Figure 2 shows a second arrangement of the failure of capture mechanisms, where the failure propagates up the fault-tree when there is a failure in both the human components AND the software components.

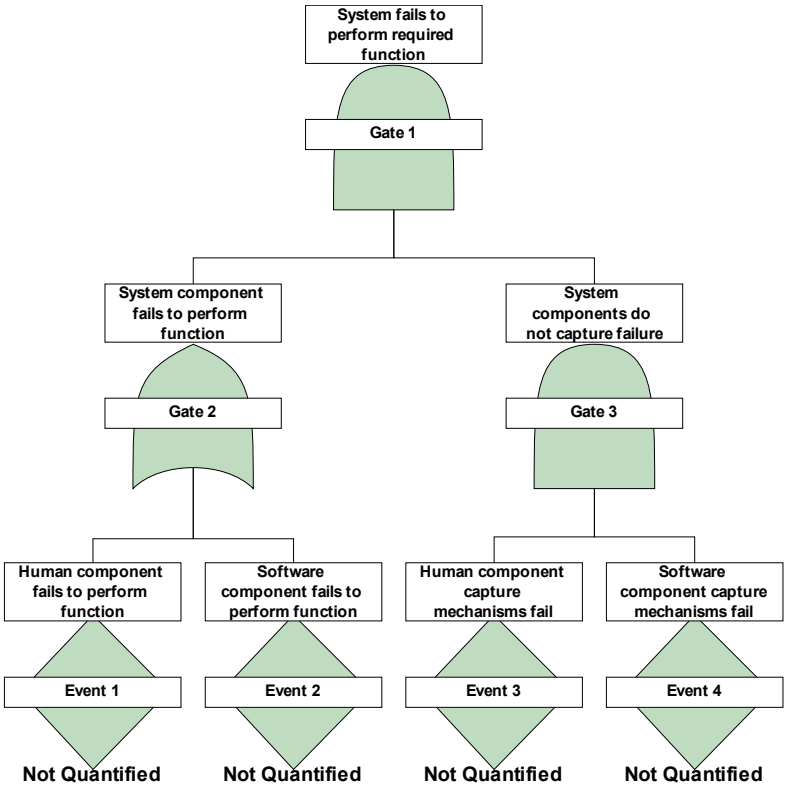

Figure 2: Second arrangement of system component capture mechanisms 'AND' gate

4. CALCULATIONS AND ANALYSIS OF THE LOGICAL INFLUENCE

Applying an example numerical set to these fault-tree events indicates the effect of the OR/AND gate swap. For the calculation comparison the error source values have been set to a nominal 0.01 per operation; the software barrier failure rate is set to 0.01; and the human barrier failure rate is set to 0.1 per operation. For the example it is not important to specify what the 'per operation' unit actually is - it could be a time based unit or an event based unit. As long as the units are consistent the calculation will be satisfied.

The logical layout of figure 1, with the OR gate in the barrier failure branch, gives the unreliability figure for the system of 0.0022 failures per operation. Figure two, with the AND gate in position, gives an unreliability figure for the system of 0.000022 failures per operation. This represents a two-order difference – very powerful.

It is fully accepted that not all systems can be represented this way. It may be said that these two positions represent the limits of reliability for a human-software system expressed in its simplest form. System designers should rightly be apprehensive of approaching the 'OR' position, and should be positively targeting the 'AND' position. This is where the power of a combined human-software system can be developed and exploited.

5. THE LIMITATIONS OF REAL SYSTEMS

As acknowledged above, the representations given in this paper do express the human-software system in one of its simplest forms. The use of the strict logic gates represents intra-system influence in its purest form – each event is mutually exclusive, as no common cause failures are present.

In reality the events expressed at a generic system level will not be specified in enough detail to guarantee no interference. In this example the sub-system components that have the software label may well have several common areas from power source to algorithm content. Similarly, the human components may well have shared frailties – in the 'OR' limit situation, they may even be the same human being.

The obvious development strategy for a human-with-software based system is to ensure that the whole system is biased towards the 'AND' condition, rather than the 'OR'. The next section of the paper will focus on strategies to promote this.

6. STRATEGIES TO PROMOTE POWERFUL RELIABILITY

The earlier definition of reliability contains two critical elements of reliability – the performance of a function and a period of time factor. Thus reliability can be said to be related to variables of a quality performance and a timely performance. It is not important (in this paper anyway) to solve the relationship completely, it is more significant to acknowledge this way of expressing reliability as a focus for strengthening overall reliability.

As contributors to reliability, humans and software have time and quality aspects, which when understood can lead to methods of improving the effects of them. Both software and humans have a finite speed of performance – with the latest technology software should easily be the faster of the two with faster reaction times [HSC 1998]. They also both have quality factors, but these are derived from different sources. Software has a consistent quality behaviour – it will always do exactly as instructed, it doesn't break or wear out. If the code instructions are incorrect for the purpose, the performance will always be likewise. The human however, has a progressive quality performance – the human can learn and respond through intellectual consideration, however humans do break and certainly wear out.

The goal of task allocation is to design a system where the tasks of the human operator are achievable and are appropriate to the operator's role (and ability), and the development of the software system is technically and economically feasible [Wright et al 1999]. In this paper there is an additional goal of promoting the 'AND' position of the failure capture mechanisms. Here each human and software operating component must give opportunity and authority for the other to review the decisions being made.

Particular strategies for software systems employ features such as systemic processor and memory self-checks, continuous testing, checksums and 'watchdogs', which contribute to the early detection of potentially unreliable failures [HSC 1998]. The human operator is then cued to review the system situation and to make a timely intervention, if appropriate.

7. CASE STUDIES IN HUMAN AND SOFTWARE CO-OPERATION

7.1 Medical Case Study

As detailed in a case study on automation, complexity and failure from the department of emergency medicine at the University of Florida [Wears 2005], an incident occurred causing great debate concerning system reliability. The incident was a patient needing emergency medication from an automated dispensing unit which had a number of software controlled safety interlocks. By coincidence on the day that the patient was rushed to the emergency room, the software interlocks were being re-enabled after a software upgrade process. This caused a storm of messages (one message for each medicine type) which slowed the system response to the extent of a complete stop. The human operators eventually were able to interpret the interface of the software and identify the failure mode, even though the screen message was unhelpful as "Printer not available". Runners were used to obtain the required drugs for the patient, and others were sourced from return bins next to the dispensing units.

The capture and correction of the problem did take time, it was due to human problem solving capabilities and a fortunate set of medical circumstances, that catastrophe was averted. The medical practitioners that had to work through this stressful situation were very critical of the (software) system reliability because the patient had nearly been lost. Hospital managers, who had the responsibility for purchasing the software system and hiring the staff, were full of praise for the (whole) system because the patient had been saved. The medical staff were considering the reliability of the system excluding themselves, and with this as the system boundary, many would agree to its unreliability. The managerial staff were looking at both components of the system, and with this as the system boundary, many would also agree to its reliability.

The timeliness of the understanding between the human and the computer interface was a critical factor in the reliability discussion. It appears this was pretty bad, but it could have been designed far better to actually promote clearer and quicker cognition of the situation. This would have promoted the 'AND' gate situation rather than the 'OR' gate position.

7.2 Automotive Case Study

A similar reliability and system boundary debate can be demonstrated through the reliable provision of driving functions in cars. Many drivers may see their cars as essentially unreliable machines. Failures arise from the mechanical components, the management software and of course the human operator. However, failure capture mechanisms are present throughout modern autos and the human computer interface is at the core. The operator relies on the engine management system to continually monitor the performance of the car. In turn the engine management system relies on the human operator (driver) to notice the cautions and warnings and take the car to the dealer or garage.

From the point of view of the automotive manufacturer, preventing propagation of sub-system failures to whole system failure enhances the reliability of the car in its driving mode. The requirement for both the human and the software to notice a developing reliability problem is promoted further by the design of the interface between the car and the driver. Specific symbols are well recognised on the dashboard, and even a key is given in the driver's manual for particular combinations. My own car's engine management system actually keeps a permanent memory of engine faults to be downloaded at the service garage, such that diagnostic time and interpretation is kept to a minimum.

Of course the economics of the whole system – automotive or otherwise, need careful review as the boundary widens. To take in more reliability enhancing capability obviously costs more money. If the system includes just the single car then the cost should be fairly low, but add in a fully manned service centre and customer care system as well, and the expense increases. The service centre costs may be considered as overhead and therefore shared among all the cars produced by the manufacturer.

The more cynical readers might be tempted to have the opinion that automotive producing companies make more profit on the servicing and maintenance functions than on the car sale in the first place, so their economics of reliability have a much different approach.

Taking the boundary of the system even further – to include the road and other road users, brings in the accident as a cause of 'unreliability'. This may not be fair as neither the driver nor the automotive manufacturer can have responsibility for the condition of the road or the ability of other drivers. However, the manufacturer can and does influence the performance of the driver – the recent addition of satellite navigation systems has led to discussion on driver distraction as a growing cause of road traffic accidents.

8. CONCLUSIONS

The design of the human-software failure capture mechanisms in both the cited examples indicates the way that the 'OR' gate and 'AND' gate positions influence the actual and perceived reliability of the system in question. Both cases also demonstrate the importance of understanding the boundary of the system and whether you, as the judge of reliability, are inside or outside of it.

The logical representations of the 'OR' and 'AND' relationship between the human and software components in the error catching operation mode, may be seen to be the limits of performance. Any system of interest will probably be somewhere in between. The closer the system approaches the 'AND' situation, the system boundary tends to increase, therefore the cost also appears to have an increasing trend.

Strategies to push the system performance towards the 'AND' condition can be promoted with due respect to operational tempo, these include;

- Deliberate cross checking between software and human functions, e.g. data format checks, controlled data entry procedures and re-presentation of data for human review.

- Functional separation of lower level tasks, with functional overlap of higher level tasks.

- Collaboration between humans and software functions for the highest reliability operations.

REFERENCES

Brain & Maguire 2005: “The Opportunity Cost of Military Equipment”, C.J. Brain MIMechE, R.L. Maguire MIMechE, SE Validation Limited, Salisbury, UK.

MoD 2004: “Human Factors for Designers of Systems – Part 15 Principles and Process”, Defence Procurement Agency, The Ministry of Defence, Kentigern House, Glasgow, UK.

Høyland & Rausland 1994: “System Reliability Modelling – Models and Statistical Methods”, A. Høyland, M. Rausland, The Norwegian Institute of Technology, J. Wiley & Sons, New York, USA.

HSC 1998: “The Use of Computers in Safety-Critical Applications – Final Report of the Study Group on the Safety of Operational Computer Systems” Health and Safety Commission, HMSO, Norwich, UK.

Wears 2005: “Automation, Interaction, Complexity and Failure – A Case Study”, R.L. Wears MD, MS. Department of Emergency Medicine, University of Florida, Jacksonville, USA.

Wright et al 1999: “Functional Allocation : A Perspective from Studies of Work Practice”, P. Wright, A. Dearden, R. Fields. Department of Computer Science, University of York, York, UK.