Task analysis for the investigation of human error in safety-critical software design: a convergent methods approach

N. M. Shryane*, S. J. Westerman, C. M. Crawshaw, G. R. J. Hockey and J. Sauer

Human Factors Group, Department of Psychology, University of Hull, Hull HU6 7RX, UK

Keywords: Safety critical systems; Task analysis; Error analysis; Human error.

An investigation was conducted into sources of error within a safety-critical software design task. A number of convergent methods of task- and error-analysis were systematically applied: hierarchical task analysis (HTA), error log audit, error observation, work sample and laboratory experiment. HTA, which provided the framework for the deployment of subsequent methods, revealed possible weaknesses in the areas of task automation and job organization. Application of other methods within this more circumscribed context focused on the impact of task and job design issues. The use of a convergent methods approach draws attention to the benefits and shortcomings of individual analysis methods, and illustrates the advantages of combining techniques to analyse complex problems. The features that these techniques should possess are highlighted.

1. Introduction

1.1. *Task analysis for the investigation of human error*

The term 'task analysis' describes a plethora of techniques intended to describe and examine the tasks carried out by human beings within a system (for a comprehensive review see Kirwan and Ainsworth 1992). The range of human factors domains to which task analysis techniques can be applied is broad, including training, task and job design, allocation of function and performance assurance. Although they have the same general goal, different techniques may be suitable for answering different kinds of questions, in different kinds of work systems. This paper is concerned with an investigation into the human factors underlying the commission and detection of human error in railway signalling software design.

Within the context of performance assurance (i.e. the consideration of factors necessary to ensure system performance within acceptable tolerances), human error is of paramount importance, especially in safety-critical systems. Many analysis methods can be used to investigate the role of human fallibility in systems (e.g. SHERPA, human HAZOP; see Kirwan 1992a). Used as Human Error Identification (HEI) techniques, they are often applied within the framework of Probabilistic Risk Assessment (PRA), where the set of undesirable events that could occur within a system is defined, along with the paths that lead to them and the probability of their occurrence. The assessment and use of Human Error Probabilities (HEPs) has been criticized when applying absolute error probabilities (Hollnagel 1993). The authors would argue that HEPs are more reliable when comparing *relative* error probabilities associated with different parts of a task than when used to give *absolute* error probabilities.

1.2. *Human reliability analysis and the nature of the system*

For process control system operation and similar environments, where many of these techniques were developed, HEI methods have undoubtedly proved to be useful. Such systems are 'physical' in nature and the hazards they present tend to be delimited, or 'bound', by the physical properties of system elements and the actual or potential physical linkages between system components. In this context an operator action—say, closing a valve—will only be able to affect aspects of the system that the valve has the physical potential to affect. Even if this potential includes causing an explosion, the sphere of influence of this is in principle determinable.

These constraints do not apply when considering the design of software systems. Errors made in the task of programming computer-based systems are not subject to the 'bounding' of error by physical principles in the way described above. Instead, the sphere of influence of an error is only limited by the power and flexibility of the programming language used. The development of complex systems requires the use of powerful languages that can express that complexity. This means that even the most simple of errors (e.g. misnaming one variable) could result in unpredictable and potentially disastrous consequences for the logic embodied by the system. These consequences will not necessarily be confined to the specific aspect of the system to which the error relates. By unintentionally overwriting a section of memory, for instance, other—logically and functionally unrelated—parts of the system can be affected. This means that virtually all aspects of the programming task could lead to hazardous outcomes, rather than particular, easily identifiable sections. As stated by Broomfield and Chung (1995: 223) 'no established technique is available for relating software faults to system hazards'.

1.3. *Human error identification methods*

The effectiveness of most HEI techniques depends upon the expertise and experience of the analyst, and this holds true for more general examples of task analysis methods as well. This is because, almost without exception, these methods are based solely or primarily upon the judgements of expert practitioners, who are themselves open to biases and errors of cognition. Whether through interviews or through task documentation, the information gathered regarding tasks and possible errors will usually be subjective in nature. Some HEI methods attempt to reduce the effects of practitioner bias by using expert-system-like computerized question and answer routines (e.g. SHERPA), but the potential problem of a task expert's faulty or incomplete mental model of the system is not addressed. This is of particular importance in the context of computer programming, where there is unlikely to be a complete mental model of how errors will affect system performance for the reasons outlined in § 1.2.

The development of a programming language, e.g. C+ + ; Ada, is a good example of a complex programming task. The development is usually accompanied or followed by a standardization process, which attempts to remove inconsistencies, undefined behaviour, etc. from the language. The standardization is carried out by committees such as ANSI (American National Standards Institute) and ISO (International Standards Organisation), made up of experts from around the globe.

However, Hatton (1995: 56) points out '[standardisation] committees simply forget things or don't understand the full implications of some features at the time of standardisation'. These errors come to light during use of the languages, and are referred back to the committee by the programmers who discover them. For the Ada language, developed in part for safety-related applications, Hatton (1995) reported that there were around 2000 of these 'interpretation requests' outstanding.

1.4. *Convergent methods approach*

It can be seen that features of the computer software design task pose problems for the investigation of human error. In the main, HEI techniques are powerful and flexible methods that can be used effectively in a range of task environments. It is contended, however, that in the study of error in computer programming, and certainly in the context of safety-critical systems, they should not be used alone. 'It is recommended that reliance is not placed on a single technique of HEI' (Kirwan 1992b: 378). A broader-based approach is needed, using more than one method in order to provide convergent validation, and to allow different parts of system performance to be adequately investigated. This calls for the use of a set of differing analysis methods and data sources, including actual task performance, rather than traditional 'expert-opinion' focused task analysis alone. Using this combination a 'matrix' of evidence regarding overall system integrity can be built up.

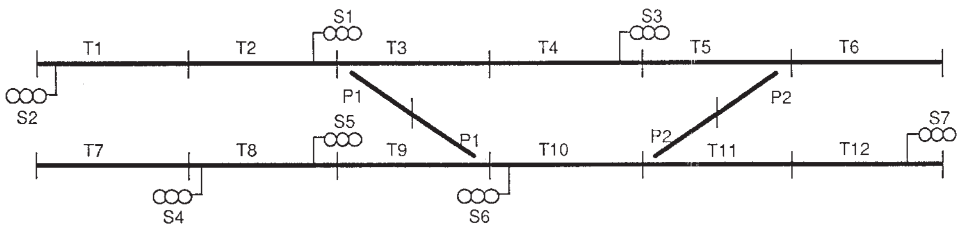
This paper considers such an approach to the analysis of human error. Specifically, the design of software for a safety-critical railway signalling control system, called 'Solid State Interlocking' (SSI), is described in § 2. Section 3 details the initial task analysis of this design process. Section 4 describes an empirical error analysis used to provide convergent evidence for the investigation. It should be noted that this study is an investigation into the factors affecting production and detection of error, not a PRA of the system.

2. Solid State Interlocking

As a case study of a complex safety-critical system, the design of data for a railway signalling safety system was investigated. 'Solid State Interlocking' (SSI) automatically controls the signals, points, etc. in a section of railway, ensuring that only safe movements of trains are allowed to take place. Each SSI is triply hardware redundant (three identical central processors), but the 'geographic data' that specifies the logic for the movement of trains is unique to each SSI, and the same version is loaded into all three processors. This means that the 'data' must be correct, as any faults could allow collisions or de-railments.

Figure 1 shows a small section of a highly simplified signalling diagram. It represents a plan view of the railway layout. Track sections are shown labelled T1, T2, etc. Signals are represented by three schematic lights and are labelled S1, S2, etc. Where tracks converge or diverge there are gaps, representing sets of points, labelled P1 and P2.

Below the diagram is a simplified section of data, showing the conditions that must be fulfilled before Route 2 (R2; from S1 to S7) can be set. This entails checking that the route is available (R2 a), e.g. not barred because of maintenance; that the points are in the correct position, or free to be moved to the correct position (P1 crf, P2 cnf); and that other, opposing routes are not already set, which is done by checking two opposing sub-routes to ensure that they are free (U10-AB f, U3-BC f). If these checks are passed, then the route is set (R2 s); the individual sub-routes in



```
*QR2   if    R2 a
             P1 crf , P2 cnf
             U10-AB f , U3-BC f
       then  R2 s
             U3-CB l , U9-CA l , U10-BA l ,
             U11-BA l U12-BA l
             P1 cr , P2 cn
             S1 clear bpull
```

Figure 1. Example SSI signalling layout and data.

Route 2 are locked (e.g. U3-CB l); the points are moved to the correct position (P1 cr, P2 cn); and the route entrance signal is checked to see if it is available to be changed to green (S2 clear bpull). It is the programming, verification and validation process for this data that is the focus of the present study.

3. Task analysis

The technique of Hierarchical Task Analysis (HTA: Annett and Duncan 1967) was chosen for the initial investigation of the SSI design process. HTA has existed for so long, and it has been so widely used, that it could be described as 'traditional'. Associated with it is a range of 'traditional' sources of data, including the technical—and the critical incident—interview, system documentation and operator observation. This technique was chosen because of its flexible and systematic task decomposition process. Used with the data sources listed above, it would provide the framework for the later error analysis, which could then be used to provide a retrospective validation of the technique.

3.1. *Hierarchical task analysis method*

A 'process analysis', as suggested by Piso (1981), was conducted concurrently with the initial stages of the HTA. The process analysis sets out to describe the production process, functioning of equipment, jargon, and general task environment, so that a framework for the HTA itself is provided. HTA is used to describe the goals that the operator must fulfil and the plans to schedule them. First, the overall, or superordinate, goal of the SSI design system is broken down into a number of sub-goals, along with the plan required for structuring, ordering and performing them. This procedure is then iterated, each sub-goal being assessed to see if it warrants further redescription. Consideration of performance shaping factors (e.g. expertise, task design, performance demands, office environment) informed the assessment of error-proneness of the operation or plan, and hence the level of redescription required.

The strengths and limitations of each of the sources of data listed earlier, i.e. interviews with task experts, task-related documentation and video recorded task observation, are discussed below.

3.2. *Sources of data for HTA*

3.2.1. *Interviews*: Interviews with task experts are an accessible and flexible method of data collection. Interviewing is, however, a purely subjective method, liable to error, omission and bias. To reduce these effects a number of steps were taken. Various personnel were interviewed, from managers to trainees, to avoid gaining a view of the data design task coloured from one organizational perspective. As well as formal interviews, informal chats with staff were entered into wherever possible. It was felt that, especially in a safety-critical industry, the less the participants were made to feel scrutinized the more likely they were to be honest about the way in which tasks were conducted and about the errors that could occur.

3.2.2. *Documentation*: Guides, manuals and training materials give detailed information on how a task *should* be done, although they are unlikely to reveal how a task is *actually* done. In the domain of programming tasks, it will tend to be more a description of the 'tools of the trade', rather than a 'recipe' to achieve correct task performance.

3.2.3. *Observation*: Task observation allows features and problems of task performance to be revealed that otherwise may not come to light using the above methods (e.g. assumed knowledge). The 'observer effect' is, however, likely to have an influence on task performance that is especially unwelcome when investigating error.

3.3. *Task analysis results*

The data gathered from all sources were amalgamated into a hierarchical diagram of the SSI data design process. For a fuller description of the HTA, see Westerman *et al.* (1994). The full diagram consisted of seven levels of the hierarchy, around 150 individual operations structured by 40 plans of various complexity. A brief overview of the process of SSI data design is shown in the hierarchical diagram (figure 2), showing the first three levels of the hierarchy.

Box 0 contains the overall goal of the system, boxes 1 and 2 below being the sub-goals required to accomplish it. The relationship between boxes 1 and 2 is shown in Plan 0. These sub-goals are then further redescribed in boxes 1.1 to 2.2. Each horizontal level of the hierarchy is a complete description of the task, at higher detail the lower the level.

The task is roughly divided into two areas (boxes 1 and 2 on the diagram): office-based production and site-based installation. The actual programming process consists of four stages: preparation (writing; 1.2), set to work (simple testing; 1.3), bench checking (code inspection; 1.4) and testing (1.5). These stages will be the focus of this study. In terms of the system life-cycle, these stages correspond to the detailed-design and build of the system (1.2, 1.3), and the verification and validation process (1.4, 1.5). Plan 1 specifies how these tasks are linked. The equipment available for the task is a dedicated UNIX system called the Design Work Station (DWS). The data are written at the terminals of the DWS. Printouts can then be produced for static code inspection, or data checking as it is called. The data can also

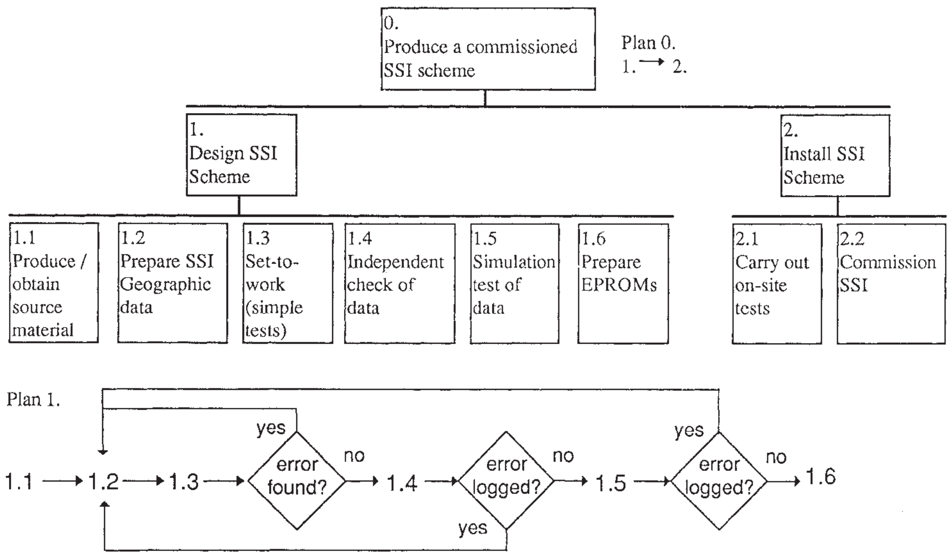


Figure 2. Overview of the HTA for designing and installing an SSI.

be downloaded to an SSI processor module, which acts as a simulator for the functional testing of the data. The tester performs functional tests of the SSI data by using a trackball to scroll across a graphical representation of the signalling layout (similar to figure 1), and setting elements to their various states.

Essentially, the SSI data are written, and then briefly tested using a computer-based simulation of the railway network. This first testing session is carried out by the author of the data, to ensure that it meets certain minimum standards. A printout of the data is then checked by another engineer who has had no part in the writing process. Finally, the data are again loaded onto a simulator, where they are subjected to more rigorous and formal testing by a specially qualified engineer who must have had no part in either of the preceding stages. If any errors are found during either the check or formal test, they are logged and the data returned to their author for amendments, and the whole cycle gone through once more.

Regarding the staffing for the various stages, expertise tends to increase from left to right. The most expert and qualified signalling engineers are employed at the testing stage, the 'last line of defence' to remove faults in the data. The least experienced are employed in writing the data, although they can seek help from more senior engineers who are not involved on the project. Expertise is gained in the first instance by a number of 2-week training courses that instruct the newly recruited trainee-engineer in basic railway signalling and SSI data. Most training is done 'on-the-job', however, with novices tackling progressively more complicated aspects of the data preparation task, and then going on to qualify for checking, etc. There is not the space here to review all of the potential problems highlighted by the HTA, but some of the more interesting ones are described below.

If the HTA is carried out systematically, structural features of the HTA hierarchical diagram can be used to highlight elements of the task being studied. For instance, the pattern of the overall diagram (not shown) and the specific goals within this pattern revealed the similarity of the writing and checking tasks compared to the testing task.

Consideration of an organizational change revealed a potentially serious problem relating to the introduction of automation in the data writing task. To increase productivity, all of the signalling firms taking part in this study were in the process of developing computer-based tools to automatically write much of the simpler, rule-based data straight from a signalling plan. This will have the effect of removing much of the training that novice data writers gain by tackling these tasks, and leave them less equipped to handle the more complex, knowledge-based data that was identified in interviews as the most problematic.

Time pressure occasionally forces some checking and testing to be carried out in parallel, leading to a revised Plan 1 (figure 3). This means that the version control for the data must be very tight, or unchecked data could be signed-off as safe by the tester. Normally, each new version of the data is given a unique version number by the data writer. This number records how many cycles of checking and testing the data has gone through, but not whether the latest version was generated because errors were found in a check or a test. If it was a test, has that version of data been checked as being error-free before? The danger point is shown by the dashed diamond in figure 3. If this decision is made incorrectly, unchecked data could be released into service. This problem is exacerbated by the contracting-out of the checking or testing of these 'rush' jobs to other signalling firms, with an attendant increase in the difficulty of version control.

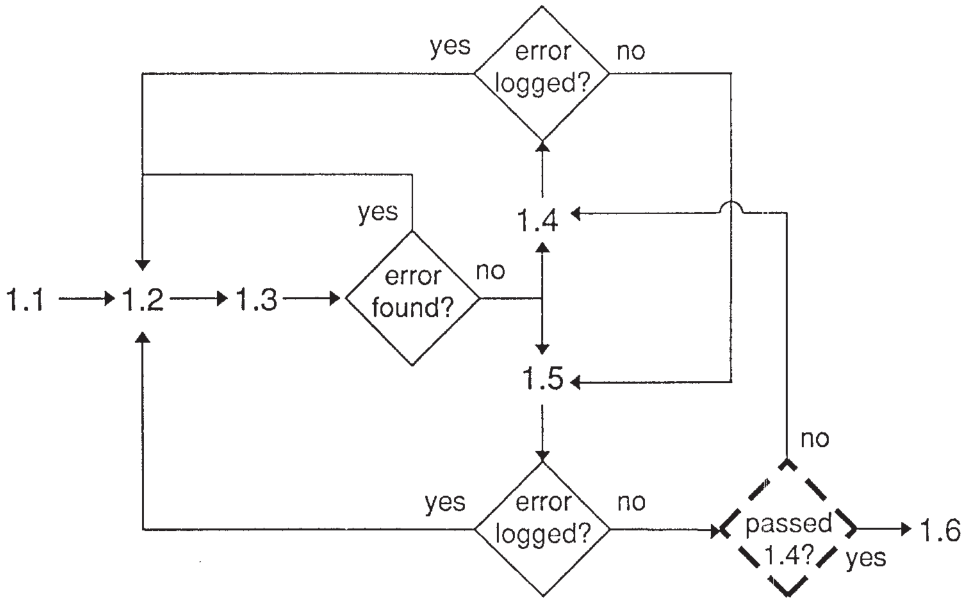


Figure 3. Revised plan 1, for when checking (1.4) and testing (1.5) are carried out in parallel.

3.4. *The need for error analysis*

The first phases of the analysis showed that HTA provides a useful framework for the breakdown of potential problem areas in a task. As discussed above, structural elements of the hierarchical diagram can be used to show similarities and differences between tasks. However, these similarities do not necessarily equate to similarities in actual task performance. For example, although identified as similar by the HTA, performance in writing and checking may not be identical even given identical data, i.e. data that are difficult to write may be easy to check and vice versa. What HTA does not reveal is how all of the variables that *may* affect task performance will *actually* combine to produce error.

4. Error analysis

The HTA was used to identify the key stages of SSI data production: writing, checking and testing. These were then analysed using a variety of empirical techniques. Crucially, not only the individual tasks in isolation, but also the combination of tasks that together make up the overall system needed to be assessed.

Several complementary techniques were chosen to provide data for the error analysis. These were chosen partly on the basis of availability, but also to give a broad range in terms of the type of data that they would provide. Existing error logs would be supplemented by observation to provide work-based data. A work sample and laboratory experimentation would be used to investigate in detail issues brought to light from the workplace. These methods are now addressed in sequence. (For further details see Westerman *et al.* (1995a).)

4.1. *Error log audit*

4.1.1. *Method*: The logs used to communicate faults found in formal checking and testing to the data writer were the data source for this method. They revealed errors made not only by the data writer (boxes 1.2 and 1.3 in figure 2), but also by the data checker (box 1.4 in figure 2), as faults found by the tester necessarily have been missed by the checker. These logs detailed the SSI data faults in terms of their functional manifestation rather than their underlying psychological cause, but still provided rich information about the nature of errors. The main strength of this method was its intrinsic validity: the errors were all committed and detected by signalling engineers carrying out their normal job of work.

The primary weakness of the method is the uncontrolled nature of the variables underlying the production and detection of errors. The number of faults in the data depends mainly upon the skill of the data writer and the complexity of the work. Usable measures of these factors were not available, however. In an attempt to reduce bias, logs were included from 12 different signalling schemes, conducted by seven different signalling firms at nine sites. Even when faults within a single scheme were compared, so controlling for expertise and complexity, the detection of a fault by the checker means that the fault is therefore unavailable for detection by the tester.

Two classes of error were not recorded in the logs. The first class is that of the errors committed by the data writer, but also detected by him or her. These would be corrected when discovered, and not be logged and passed on to the later stages. The second class is that containing faults that are not caught by either checking or testing—perhaps the most important to study. Information regarding any faults discovered after the data were installed on-site, or any faults that remained latent until accidents or incidents, was not available.

4.1.2. *Results*: Table 1 shows the breakdown of 580 faults detected by checking and testing, 290 from each. The 'T/C ratio' specifies the ratio of faults detected by testing versus checking within each category (i.e. testing divided by checking). The higher the figure, the greater is the frequency of detection by testing compared to checking. The nature of the information in the logs means that the fault categories mostly relate to the railway signalling principle that they would violate. The exceptions are the 'None' category, which relates to false-alarm errors by the checker or tester (the data were actually correct); and the 'Other' category, which details miscellaneous SSI data-language specific errors, which do not relate to specific signalling principles.

There are a number of reasons why it would be misleading to use these results to make a formal quantitative comparison of the relative efficacy of the checking and testing processes. First, the error logs were gathered from a number of different schemes and therefore the checking and testing logs were not fully matched. Second, any faults detected at the checking phase were not available for detection at the testing stage, and consequently there is no means of estimating the efficiency of the testing process in detecting these faults. Third, there were no measures available of the total numbers of faults that escaped both checking and testing, again making estimation of the efficacy of the testing stage problematic.

Given these reservations, a number of qualitative features are of note, however. The T/C ratio shows an advantage of checking over testing for the 'Other' category. The 'Opposing locking' category, which contains errors made in barring the setting of two conflicting routes, shows a bias towards testing. Also of note is the preponderance of faults in the 'Identity', 'Other' and 'Route' categories (62.8 % of all faults). Much of the SSI data that relates to these categories is similar to that shown in figure 1. Overall, it is simpler and more straightforward than data controlling other functions, and could be considered as requiring more skill- and rule-based performance and less knowledge-based performance (Rasmussen 1983) than data relating to 'Aspect sequencing', 'Opposing locking', etc.

A total of 12.4% of all faults logged turned out to be false alarms, where no actual fault was present in the data. Several 'repeat' faults were found in the logs. These refer to faults found at either the check or test that were still present when the

Table 1. Faults detected at the checking and testing stages of SSI data production.

Fault category: signalling principle	Checking	Testing	T/C ratio	Category (% of total)
None (false alarm)	46	26	0.57	12.4
Identity and labelling errors	26	30	1.15	9.7
Route setting	74	120	1.62	33.4
Signal aspect control	14	32	2.29	7.9
Approach locking	11	24	2.18	6.0
Opposing locking	9	39	4.33	8.3
Aspect sequence	4	12	3.00	2.8
Other	106	7	0.07	19.7
Total	290	290	1.00	100

SSI data were re-checked or re-tested, and testifies to difficulties in writing the data correctly. These faults applied to particularly novel, knowledge-based SSI data. Where information was available to show the number of faults logged at successive checking cycles, seemingly simple faults made in the 'Identity' and 'Other' categories were the only ones to escape detection until the fourth cycle. Similar information was not available for the testing stage.

4.2. *Error observation*

4.2.1. *Method*: Errors committed but subsequently detected by the same writer, checker or tester were not recorded in the error logs. To compensate for this, a period of *in situ* error observation was conducted. In addition to the task areas accessed by the error logs, this method provided some insight into the fallibility of testers (box 1.5, figure 2). Different engineers were video recorded performing several hours of the three main task areas. Interference of the task could affect error-rate, so to disturb the tasks as little as possible the participants were asked to carry on their work as normal, but to comment if they found that they had, or had nearly, made an error. They were then prompted as to the reasons they identified as causing the error, in an attempt to classify it as a skill- or rule-based slip or lapse, or a knowledge-based mistake (Rasmussen 1983; Reason 1990). The complexity of the task being undertaken, and the expertise and experience of the engineer were also used to inform the categorization of errors. While the possibility of 'faking' a lack of errors existed, the video recording of the sessions and possibility of peer review by other engineers effectively minimized this.

4.2.2. *Results*: Table 2 shows the number of errors flagged by participants during the periods of task observation shown, and is divided up by task stage. The writing task is further subdivided by the nature of the specific writing task being carried out. Data writing does not equate to mere typing (editing), but also includes reviewing, checking specifications and requirements, and planning. It can be seen that data editing on its own accounts for most of the errors in the writing task as a whole. The most error-prone signalling principle in the data editing was 'Opposing locking', with five errors. Of these, the most time consuming of all of the errors was observed. Fully 1.5 hours were spent by one engineer attempting to find the relevant help in the paper-based manual available for the task. In the testing stage, most errors related to confusion over elements selected for testing on the VDU display. No knowledge-based errors were observed in any stage. No errors at all were observed in 3 h of checking.

Table 2. Errors observed at the writing, checking and testing stages of SSI data production.

Error category	Writing (all-including editing) (7.5 h)	Writing (editing only) (3 h)	Checking (3 h)	Testing (6.25 h)
Skill- and rule-based errors	31	25	0	5
Knowledge-based errors	0	0	0	0
Errors per hour	4.1	8.3	0	0.8

4.3. *Work sample test*

4.3.1. *Method*: Since the above two techniques are based on naturalistic analyses of work, they are susceptible to a lack of control over the initial SSI data. To compensate for this, a data writing work sample test was devised that would further inform the analysis of box 1.2 of the HTA (figure 2). A previously completed data set (for an SSI currently in service) had sections of data removed, and signalling engineers were recruited to complete the work under controlled, but work-based, conditions. This enabled task factors (e.g. complexity) to be studied. Fifteen data writers took part in the test. Although this is a small number for statistical purposes, it represents approximately one-third of all of the suitable candidates working for organizations participating in the research project. The participants in the work sample test had a wide range of experience, from 6 months to 10 years, and were drawn from three different signalling offices. To ensure the representativeness of the work sample a highly experienced engineer was employed to select the SSI data, which included all of the main aspects of the writing task. The input of the researchers was used to ensure that both straightforward and novel SSI data elements were represented, so that rule-based and knowledge-based performance could be assumed to be utilized by the participants.

4.3.2. *Results*: Table 3 shows the completion times and faults for the different task types used in the test. For the rule-based task performance, faults in the completed data were scored per signalling function violated, as for the error log audit. This avoided the biasing of the results possible when one cognitive error resulted in many faults in the SSI data. For instance, one error made by a number of engineers, involved the unintentional omission of a whole signalling function. This single cognitive error resulted in the omission of several lines of data and tens of data 'words'. Assessing errors by the line or data 'word' would have given arbitrary scores dependent on how many lines or words made up a particular signalling function.

This scoring system would have been inappropriate for the knowledge-based data, as the small amount of code involved related to only one signalling function, which nobody got completely correct. Additionally, each data 'word' in the knowledge-based section contributed a particular aspect to the overall functionality of the feature, independently of the other 'words' (which was not generally the case for the rule-based data). The knowledge-based task element was thus scored by the data word, to give a usable index of 'correctness'.

Table 3. Completion times and data faults in the work sample test.

	Completion time (s)			Faults	
Aspect of the task	Mean	SD	Time per line	Mean	SD
Rule-based performance (124 lines of data; n= 15)	9518	2958	76.8	9.5	7.6
Knowledge-based performance (4 lines of data; n= 13)	1120	458	280	3.7	1.8
Total performance (128 lines of data; n= 13)	10103	3013	78.9	10.9	4.4

The differences between the two scoring systems makes formal statistical comparison between rule- and knowledge-based fault performances misleading. However, were the rule-based faults to be scored by the data word this would inevitably increase rather than decrease the scores, and so this points to a preponderance of rule-based over knowledge-based faults. On the other had, it can be seen that knowledge-based data is much more time-consuming to complete, per line of data, than the rule-based section (t (12) = 10.7, $p < 0.001$), and its difficulty is also demonstrated by the fact that the two least experienced participants could not complete it. (Their data are thus not included, giving $n =$ 13 for the knowledge-based elements.)

Regarding rule-based data, the problem of 'common-mode failure' was highlighted. A term usually used in system safety, it refers to situations in which seemingly independent components can fail due to the same underlying cause. Evidence for common-mode failure was found when it was seen that four specific faults were made identically by seven participants or more; indeed, one of these faults was made by 13 participants. These faults made up the largest single rule-based category, encompassing 40 of the 142 errors.There was no pattern relating to the signalling firms or sites of the participants making these errors, or their expertise. Task-related factors, and their interaction with human cognition, are thus implicated. It was found that three of the four common-mode errors related to familiar data that, because of the specific instances used, required infrequently needed parameters. These errors can be characterized as habit intrusions, or 'strong but wrong' errors (Reason 1990).

4.4. *Laboratory experiment*

4.4.1. *Method*: In real work, the detection of a fault by a checker renders this fault unavailable to a tester. Therefore, to investigate the relative efficacy of the two processes, computer-based, simplified task simulations were developed for completion by novice participants acting as either checkers ($n =$ 13) or testers ($n =$ 27). Sets of identical, detectable faults were seeded into both task simulations (16 faults per simulation), and performance of the novice checkers and testers compared (corresponding to boxes 1.4 and 1.5 in figure 2). The fault types were chosen from actual logs in the error log audit. 'Route setting' was chosen because it contained frequent faults. 'Signal aspect control' and 'Opposing locking' showed great differences between checking and testing. Two categories of 'Opposing locking' faults were chosen, reflecting the two different methods by which this signalling principle is dealt with in the SSI data. It was thought that these two methods may have differential effects on the relative efficacy of checking versus testing. Some fault categories (e.g. 'Aspect sequence', 'Other') could not be used because of the need to make the errors equally 'visible' to checkers and testers, and the simplification of the simulation compared to the real task.

4.4.2. *Results*: Table 4 shows the probability of detection of the four types of faults seeded into the checking and testing task simulations. Analysis of variance revealed no main effect of task type (checking versus testing), but there was a significant interaction between task type and fault type (F (3,72) = 10.58, $p < 0.001$). This is attributable to the comparatively poor performance of checkers in detecting 'Opposing locking I' faults, and of testers in detecting 'Signal aspect control' faults.

Table 4. Probability of fault detection (checking versus testing) in SSI task simulation.

Fault type (4 faults per category)	Checking	Testing
Opposing locking I	0.69	1.0
Opposing locking II	0.83	0.89
Signal aspect control	0.94	0.41
Route setting	0.90	1.0

False alarm rate (i.e. the percentage of logs relating to non-existent faults) was 22.5% in the checking task, 27.6% in the testing task, 25.0% overall.

5. Discussion

The utility of different task- and error-analysis techniques, as applied to the identification and analysis of human error in the SSI design process, is apparent when these methods are considered in concert. This 'triangulation' (Denzin 1988) confers powerful cross-validation of the techniques by consideration of how the evidence presented by each combines when applied to the same task area. It may be that the methods do not address the same factor, in which case the methods are independent. If they do address the same factor, they can either converge or diverge.

5.1. *Error type*

An example of divergence between analysis methods is seen when considering the type of error most problematic for the SSI data production system. HTA critical-incident interviews consistently identified complex, knowledge-based data design tasks as the most problematic type. In support of this, no participants in the work sample were able to complete the knowledge-based data without faults, and the time taken to complete this small amount of data also testifies to its difficulty. In the error logs, novel, complex data were repeatedly passed on to the checking stages while still incorrect, again suggesting the difficulty in writing these data. The error observation technique showed that knowledge-based errors also appeared much less likely than rule-based errors to be self-detected during writing.

On the other hand, the error log audit found that the majority of faults were being discovered in simpler, rule-based SSI data. From the error observation it was seen that around eight skill- and rule-based errors per hour were being self-detected while data editing, and the work sample test showed that around four skill- or rule-based errors per hour were not being self-detected. This indicated that a large number of simple errors were being committed in the data writing, with a significant proportion not being self-detected by their engineer. Although a comparison of errors per hour between faults in the rule- and knowledge-based data in the work sample would not be fair (for the reasons outlined in § 4.3.2), it was shown that, as in the error logs, many more rule-based faults occurred over the whole task.

So knowledge-based data seemed noticeably difficult and error prone to write, but in all tasks viewed and the error logs, rule-based errors and faults predominated. The answer probably lies in the relative opportunities to commit both types of error. The work sample task contained only a small amount of complex, knowledge-based data to complete, but it was selected in part because of its similarity to actual work; indeed, it was part of an actual SSI scheme.

A factor not considered in the above argument is fault detection. The error logs do not provide an accurate ratio of knowledge-based to rule-based faults pre- and post- checking and testing, which would allow the relative detectability of each to be known. However, there is some continuous data (i.e. from the same SSI dataset) relating to faults detected at each stage of checking. This showed that the only faults to pass through three cycles without detection were ones in simple, rule-based data. Again, this may not necessarily be because simpler faults are actually any harder to detect than the knowledge-based ones. Indeed, even if they are easier to detect, the number of simple faults escaping detection may be higher because of their higher prior probability in the data.

Another factor in the number of rule-based faults may be the relative lack of emphasis placed on their importance, as shown by the HTA interviews. This would lead to more attention being given to parts of the data seen as challenging or potentially more error prone than the simple and straightforward aspects. However, virtually any faults in the data could have disastrous consequences.

The HTA suggested that knowledge-based data writing was more complex and made more demands on the operator than simpler data, and this was supported by the work sample test. However, HTA has little provision to represent explicitly the event-rates that would have shown that knowledge-based data, although perhaps more error-prone, is written far less often than the simpler data, and so has correspondingly less influence on overall system integrity.

The prevalence of simple errors and their difficulty in detection has unanticipated consequences for the introduction of automation. Although the original reasons for the development of automation was economic, its introduction may have greater than expected benefits if it can eradicate a potentially dangerous source of data faults. However, its implications for training will still need to be addressed.

5.2. *Common mode error*

Common mode error was another problem area uncovered by the study. Checking, or static code inspection, is seen as the most efficient method for revealing error in computer programming code (Bezier 1990). The principle reason for this is that the whole of the code can be inspected, as opposed to the limited amount of functionality that can usually be tested in complex systems. (The error logs and experiment showed both checking and testing to be similarly efficient; but this is probably because a greater proportion of SSI functionality is testable than for normal programmes.) However, the similarity of checking to the writing stage, suggested by the HTA, may reflect underlying similarities in the tasks and mental processing required by both stages. If this is the case, then checking may suffer the same weaknesses as writing, and so the processes may be liable to common-mode failure. Indeed, a fault type seen to be problematic in the data writing error observation ('Opposing locking'), was detected poorly by checkers in the error log audit and in the laboratory experiment. Further evidence for this similarity is also provided by the fact that the kind of simple errors observed most frequently in the observation were those most resistant to detection by checking in the error log audit. This may seem odd when considering simple slips, as they can be generated by processes not affecting the checkers (e.g. typographic errors), and they are also easiest to self-detect. However, the most striking examples of common-mode error were the identical errors committed in the work sample test, with up to 87% of the engineers making exactly the same error. This highlights the vulnerability of even highly trained engineers to exactly the same errors of cognition when in the same task environment. Checkers will be liable to these same errors because of the similar task factors to writing, and self-detection of these 'habit intrusions' will not be good, as they are errors of intention, not action. Common-mode errors were also seen in the laboratory experiment, with task environment (checking or testing) effectively predicting which types of detection errors would be made.

5.3. *Task diversity*

The converse of task similarity is task diversity. Differences between checking and testing, suggested by the HTA, were confirmed by qualitatively different error detection performance between checking and testing found in the error log audit. While encouraging, the error log results may have been due to a number of factors other than the task environment (e.g. the initial number and type of faults in the data). However, the result was confirmed in the laboratory experiment, where the seeded faults were exactly the same for checkers and testers. The results showed that while there was no difference in the overall fault detection performance between the two methods, they did lead to the discovery of different types of fault. Almost inevitably, because of the differences between the real tasks and the simulations, there were some differences of detail between the error-log and laboratory experiment results, e.g. in Signal aspect control faults. A number of factors may have contributed to this, e.g. the difference in participant characteristics, the reduced range of faults in the simulation (16 versus 580 in the error logs). Given these differences, however, the fact that both studies still showed different fault detection characteristics between checking and testing indicates that rather than the two stages being an example solely of redundancy in the system, they are instead an example of task diversity.

The use of task diversity can have positive implications for fault detection tasks (Fagan 1986). Different task representations are likely to engender different mental models in the operators, and lead to different emphases and task performance strategies. This can render people less vulnerable to the threat of common mode error present when tasks are too similar, as shown between SSI data writing and checking. This diversity of knowledge, strategy and mental model of a task is known as 'Cognitive diversity' (Westerman *et al*. 1995b, 1997).

The diversity between checking and testing has implications for the consequences of the check stage being missed out when carrying out checking and testing in parallel (figure 3). This is because some of the faults in the data, such as those that fall into the 'Other' category, are less likely to be detected by testing alone. To help to ameliorate this problem a computer-based logging tool is currently under development by the research team, to assist with fault logging and data version control. It will also record some of the *psychological* error mechanisms that lead to the faults, and help to aid the identification of further error reduction techniques.

5.4. *'Opposing locking' faults*

A consistent difference between checking and testing performance was found for 'Opposing locking' faults. It was apparent that checkers found these faults more difficult to detect than testers, so a number of further experiments were performed to ascertain why this might be the case (Shryane *et al*. 1996). 'Opposing locking' is dealt with in SSI data by the use of sub-route labels, which define a section of track and also the ends of this section that the train will enter and leave by. For example in figure 1, 'U3-CB' specifies the sub-route 'U3' (corresponding to track section 'T3') and that a train will enter/exit in the 'CB' direction. Exactly what 'CB' means in terms of the associated spatial configuration of the signalling diagram is determined by the particular layout of the track, and is different for different track section shapes. It was found that this inconsistent mapping between the spatial information of the diagram and the textual sub-route label was associated with poor performance in a simulated checking task. Testers do not have to assess the sub-route labels directly, only the signalling functionality associated with their action.

5.5. *Convergent methods*

The methods used in this analysis all had a role to play in the investigation of human error. In the first instance, HTA provided an overview of the task not afforded by the other methods. Additionally, HTA does not have to be a study of the task *as is*, but can be used to investigate the implications of variations in the task. Together, these features allowed the discovery of the organizational issues regarding the introduction of automation. It is also useful for the consideration of sequencing and scheduling of tasks, by nature of its plans, leading to the discovery of problems with parallel checking and testing (figure 3). These issues were not brought to light by the other methods. HTA's weakness was found to be in part due to the subjective nature of its data sources. However, the inability to show quantitative aspects of tasks, such as event and error rate, and how this affects the system is the biggest drawback of HTA when considering the study of error.

The error log audit and error observation are both essentially error sampling procedures of the task. Although the information that they produce can be 'noisy' due to its work environment origin, they do provide quantitative data on event and error frequencies. The error logs did not provide direct evidence of certain types of errors, however (e.g. self-detected errors), and so to compensate for this in the current study, error observation was used. The error observation seemed to be useful in recording errors in relation to overt actions, rather than to covert cognitive processes, as shown by the lack of knowledge-based errors while checking. 'There is some evidence . . . that while people are good at catching their own errors of action, they are much less good at catching their own errors of thinking, decision making, and perception' (Senders and Moray 1991: 78).

Error observation was useful for pointing out interface and task support factors. The errors in the data editing part of the writing task point to the error proneness of inputting data through a standard keyboard, which may be reduced by the use of visual programming environments and direct manipulation of code. However, this would then make the data writing task environment more similar to the testing task, so reducing overall system diversity. The system-wide effects of such factors need much further investigation. The inadequacy of existing manuals, the primary reference when needing assistance, was also observed. As a further part of this project, a computer-based 'help' application is being developed. This will include the information contained in the existing paper-based manual, but with improved searching and cross-referencing, and the ability to annotate and personalize the manual to support individual working styles. Job support tools such as these may help to reduce the problem of less training, but again research is needed to see to what extent.

The work sample test and laboratory experiment are both types of experiment. These were needed to study the variables identified as important by earlier stages, e.g. task diversity. Although less controlled than experimentation in the laboratory, the work-based version is more valid in terms of environment, task and participants. However, when studying task-related variables for which task knowledge and experience may matter, the use of naïve participants in the laboratory may be beneficial.

6. Conclusions

From the evidence presented above, two dimensions of variation can be identified with respect to techniques used for the investigation of human error in this study. First, the techniques varied in their capacity to represent event—and therefore error—frequencies; HTA lacks the capacity of the other, more empirical, techniques in this respect. Second, the empirical techniques differ in the familiar trade-off between validity and control. Error logging and observation represent highly externally valid techniques. Laboratory experimentation represents the extreme of control and internal validity, with work sample tests offering characteristics between the two ends of the spectrum. Used here to investigate human error in safety critical systems, analysis of human error will in any work-based system benefit from the application of techniques that vary in these properties.

Acknowledgements

The work described here forms part of the project ‘Human Factors in the Design of Safety Critical Systems’ (Grant No. GR/J17319) from the UK Safety Critical Systems Research Programme supported by the Department of Trade and Industry and the Engineering and Physical Sciences Research Council. It was completed with the collaboration of Interlogic Control Engineering, GEC Alsthom Signalling Ltd, Signalling Control UK Ltd and Westinghouse Signals Ltd.

References

Annett, J. and Duncan, K. D. 1967, Task analysis and training design, *Journal of Occupational Psychology*, **41,** 211 – 221.

Bezier, B. 1990, *Software Testing Techniques* (Amsterdam: van Nostrand Reinhold).

Broomfield, E. J. and Chung, P. W. H. 1995, Using incident analysis to derive a methodology for assessing safety in programmable systems, in F. Redmill and A. Anderson (eds), *Achievement of Assurance and Safety. Proceedings of the Safety-Critical Systems Symposium,* Brighton, February, (London: Springer-Verlag).

Denzin, N. K. 1988, *The Research Act: A Theoretical Introduction to Sociological Methods,* 3rd edn (Englewood Cliffs, NJ: Prentice-Hall).

Fagan, M. E. 1986, Advances in software inspections, *IEEE Transactions on Software Engineering*, **12,** 744 – 751.

Hatton, L. 1995, Programming languages and safety related systems, in F. Redmill and A. Anderson (eds), *Achievement of Assurance and Safety. Proceedings of the Safety-Critical Systems Symposium,* Brighton, February, (London: Springer-Verlag).

Hollnagel, E. 1993, *Human Reliability Analysis: Context and Control* (London: Academic Press).

Kirwan, B. 1992a, Human error identification in human reliability assessment. Part 1: Overview of approaches, *Applied Ergonomics*, **23,** 299 – 318.

Kirwan, B. 1992b, Human error identification in HRA. Part 2: Detailed comparison of techniques, *Applied Ergonomics*, **23,** 371 – 381.

Kirwan, B. and Ainsworth, L. K. 1992, *A Guide to Task Analysis* (London: Taylor & Francis).

Piso, E. 1981, Task analysis for process-control tasks: the method of Annett *et al.* applied, *Journal of Occupational Psychology*, **54,** 247 – 254.

Rasmussen, J. 1983, Skills, rules and knowledge; signals, signs and symbols, and other distinctions in human performance models, *IEEE Transactions on Systems, Man and Cybernetics*, **SMC 13**(3), 257–266.

Reason, J. 1990, *Human Error* (Cambridge: Cambridge University Press).

Senders, J. W. and Moray, N. P. 1991, *Human Error: Cause, Prediction, and Reduction* (Hillsdale, NJ: Lawrence Erlbaum).

Shryane, N. M., Westerman, S. J., Crawshaw, C. M., Hockey, G. R. J. and Wyatt-Millington, C. W. 1996, The influence of task difficulty on performance in a safety-critical labelling task, in A.F. Özok, and G. Salvendy (eds), *Advances in Applied Ergonomics* (West Lafayette, IN: USA Publishing).

Westerman, S. J., Shryane, N. M., Crawshaw, C. M. and Hockey, G. R. J. 1995a, Error Analysis of the solid-state interlocking design process: Report no. SCS-04, Department of Psychology, University of Hull.

Westerman, S. J., Shryane, N. M., Crawshaw, C. M. and Hockey, G. R. J. 1997, Engineering cognitive diversity, in T. Anderson and F. Redmill (eds), *Safer Systems. Proceedings of the Fifth Safety-Critical Systems Symposium,* Brighton, February, (London: Springer-Verlag).

Westerman, S. J., Shryane, N. M., Crawshaw, C. M., Hockey, G. R. J. and Wyatt-Millington, C. W. 1995b, Cognitive diversity: a structured approach to trapping human error, in G. Rabe (ed.), *Proceedings of the 14th International Conference on Computer Safety, Reliability and Security, Belgirate, Italy,* October (London: Springer-Verlag), 142–155.

Westerman, S. J., Shryane, N. M., Sauer, J., Crawshaw, C. M. and Hockey, G. R. J. 1994, Task analysis of the solid-state interlocking design process, Report no. SCS-01, Department of Psychology, University of Hull.