Human Fallibility

Further reading:

On a Wing and a Prayer: Exploring the human components of technological failure. D. Smith. Systems Research and Behavioural Science, 1992.

Human error: models and management. J. Reason. British Medical Journal. 2000.

Human error and the design of computer systems. D. A. Norman.

All of these are supplied as handouts.

Socio-technical systems

- Recall that a socio-technical system includes:
  - Computers
  - Software
  - People
  - Organisational processes and procedures
- The dependability of a S-T system is not just a technical issue but depends on the dependability of people working alone and in teams
- A high percentage of system failures are known to be the consequence of human fallibility - the known characteristic of people that they make mistakes

It is estimated that around 60% of aircraft accidents are caused or partially caused by human fallibility. Similarly, a high percentage of patients that are treated in hospital (around 15%) are there because of the fallibility of medical staff.

These accidents are often attributed to be the result of 'Human Error'. I prefer to avoid this term as so-called human error is sometimes the result of poor system design where the designers simply expected too much of the users of the system. If we do not take into account of the way in which people think, then we are likely to design systems that lead to errors.

Having said this, there are accidents which are simply a consequence of human stupidity. It is impossible to design practical systems that can handle completely stupid human behaviour. For example, an air crash in Russia was caused by the pilot allowing his son to play with the controls on an aircraft in flight. It would be extremely difficult to design a safe system that coped with such utterly stupid behaviour.

Human error

- People are fallible
- Human error a problematic term
  - Tends to be used when 'scapegoating'
  - Can cover a multitude of sins
- Human activity in the systems development process can be thought of as ***individuals*** working in isolation or as part of ***social groups or teams***, within a particular ***organisational context***
- A focus on these three areas can lead to a better understanding of process ***vulnerabilities*** to error, and the ***defences*** which can be put in place against them

One of the problems with system design has been that so-called 'human factors' has simply focused on the individual and has ignored the social and organisational environment in which the individual is working. This has two consequences. Firstly, it misses some very important vulnerabilities that can result in accidents and incidents. Secondly, it suggests that (somehow) we can solve the problem of human fallibility be this through better training, incentives, punishments, etc. This is impossible and often efforts to address the problem actually make the overall situation worse.

A common situation is where an individual stops caring about the quality of their work because the organisation has applied some sanctions to them for a 'mistake'. They then make more mistakes but make sure that they don't get caught.

Human fallibility and dependability

- Human fallibility can influence the dependability of a critical system:
  - During the development process
  - During the deployment process
  - During the maintenance/management process
  - During the operational process
- Errors made during development, deployment and maintenance create vulnerabilities that may interact with errors during the operational process to cause system failure

Mostly the discussion here is of more relevance to the operational process but it is worth remembering that paying attention to human fallibility in development could mean that better systems are developed.

Human problems in the maintenance process are particularly common. It is estimated by Sun that more than 90% of their server down time is a result of system management problems. Hardware (3%) and software (7%) problems are negligible by comparison.

Operating under stress

- As people are placed under stress, then their ability to work without making mistakes is reduced
- A common cause of stress in system development and operation is time pressure
  - If people are required to carry out a task too quickly they will make mistakes
  - These can occur during development (we must deliver the system next week), system management (we cannot take the system down for more than 20 minutes) and operation (the number of aircraft in a sector increases so a controller has less time to deal with each aircraft)
- A failing system is a stressful situation for the system operator so once failures start, they have a tendency to generate more failures
  - If the system has alarms, the presence of these alarms is a distraction to the operator
  - Dealing with the failure and doing normal work increases the time pressure

Time pressure in systems development is increasing because of increased competition. Premature release of web sites with many bugs are common.

A serious nuclear accident that occurred in the USA was made worse by alarm overload where operators had to deal with over 100 separate alarms. This increased workload and stress so they missed the two critical alarms which would have helped them diagnose the cause of the problem in the plant.

Vulnerabilities and defences

- As with other types of failure, failure-based analysis can consider the issues of human and social factors
- Vulnerabilities
  - Identify the areas where human error can lead to system failure
- Defences
  - Put defences in place that avoid, tolerate or recover from human error
- When an adverse event happens, the key question is not ‘whose fault was it’ but ‘why did the system defences fail?’

Defensive layers

- Computer-based critical systems should have many defensive layers:
  - some are **engineered** - alarms, physical barriers, automatic shutdowns,
  - others rely on **people** - surgeons, anesthetists, pilots, control room operators,
  - and others depend on **procedures** and administrative controls.
- In an ideal word, each defensive layer would be intact.
- In reality, they are more like slices of Swiss cheese, having many holes- although unlike in the cheese, these holes are continually opening, shutting, and shifting their location.

Problems arise with defensive layers because of violations where people do not follow procedures and administrative controls. However, these are sometimes badly designed so that normal use of the system is practically impossible so the operators have no option but to violate these procedures if they are to get work done.

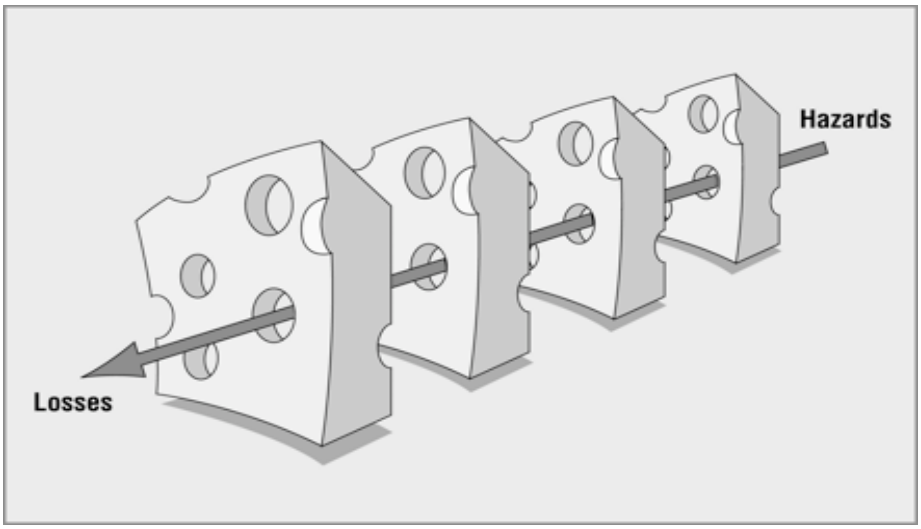
A common problem in organisations is to revise their procedures after a problem has occurred without considering how this affects the normal work of the people in the organisation. If a procedure is expensive in terms of time and resourcesl, it may never be followed.

Accident trajectories

- Accidents or critical incidents rarely have a single cause. Generally, they arise because several events occur simultaneously
  - Loss of data in a critical system
    - User mistypes command and instructs data to be deleted
    - System does not check and ask for confirmation of destructive action
    - No backup of data available
- An accident trajectory is a sequence of undesirable events that coincide in time, usually initiated by some human action. It represents a failure in the defensive layers in the system

The trigger for an accident does not have to be a human error. It could be a system fault as occurred in the Ariane 5 launcher accident. Here, the defence barriers were various checks that should have been in the software but which were not present.

Reason's Swiss Cheese Model



James Reason was a Professor of Psychology at Manchester University who studied human error and who proposed the so-called Swiss Cheese model of accidents. Each defensive layer in the system has a number of holes in it (like a slice of Swiss cheese) that represent vulnerabilities. These holes are not static (ie they are not in the same position all of the time) and, when they happen to line up then an accident trajectory has been created.

Holes are not static for several reasons. They may represent individual human actions - so a human error will be represented as a hole in this model. They may represent procedural failings that may only arise under certain environmental circumstances (e.g. when the team of people involved in carrying out a procedure is under-strength). They may represent transient software faults that are corrected automatically when the system state is re-computed.

Active and latent failures

- Active failures
  - Active failures are the unsafe acts committed by people who are in direct contact with the system (slips, lapses,mistakes, and procedural violations).
  - Active failures have a direct and usually short-lived effect on the integrity of the defenses.
- Latent conditions
  - Fundamental vulnerabilities in one or more layers of the socio-technical system such as system faults, system and process misfit, alarm overload, inadequate maintenance, etc.
  - Latent conditions may lie dormant within the system for many years before they combine with active failures and local triggers to create an accident opportunity.

Latent conditions can create long-lasting holes and weaknesses in the defences. Examples are untrustworthy alarms and indicators, unworkable procedures, design and implementation errors, etc.

Dealing with these latent conditions may lead to active operator error through time pressure, fatigue and inexperience.

Incident reduction

- Reduce the number of latent conditions in the different layers of the system (plug the holes)
  - If the number of faults in a software system is reduced, this increases the strength of the defensive layer
  - However, this technical approach ON ITS OWN cannot be completely effective as it is practically impossible to reduce the number of latent conditions in the system to zero
- Increase the number of defensive layers and hence reduce the probability of an accident trajectory occurring
- Reduce the number of active faults that occur

As well as incident reduction, organisations may also have strategies based on error recovery so that, if a disaster occurs, then there are mechanisms in place which allow relatively rapid recovery from this. Error recovery relies on error containment so that the consequences of the error are limited (in time and space).

Active fault reduction

- Understand the tasks that people do when interacting with a system
- Recognise that, within these tasks, there is a potential for human error
- Design the system for:
  - Error avoidance
  - Error detection
  - Error recovery

Again, we see an instantiation of the three principal approaches to critical systems development - avoidance, detection, recovery. These are fundamental to all work on critical systems.

Types of individual tasks

- Based on work in cognitive psychology by Rasmussen who developed the so-called s-r-k framework to classify work tasks:
  - Skill-based: routine, automatic tasks in familiar settings
    - Initiating a program, logging in
  - Rule-based: applying pre-packaged solutions to problem solving tasks
    - Installing and configuring a program, making some computation using a spreadsheet
  - Knowledge-based: tackling unfamiliar problems in unfamiliar settings
    - Designing a spreadsheet, debugging a program
- Cognitively, we tackle these tasks in different ways so make different types of error

Obviously, many real tasks don't fit neatly into one of these categories but involve elements from all of them. Therefore, if you are preparing slides for a lecture using Powerpoint, there is a skill element (inputing text), a rule element (formatting a slide) and a knowledge element (thinking what to put on the slide).

Individual errors

- Reason developed Rasmussen's task classification to create a generic model for human error:
  - Skill-based slips and lapses - **errors of action**
  - Rule and knowledge-based mistakes - **errors of intention**
- Skill-based slips - we know what to do but do the wrong thing
  - Login using the wrong password
  - Plug a cable into the wrong socket
- Rule-based errors - we fail to chose the right rule or violate rules
  - Miss out a stage in the configuration process
  - Enter numbers at the wrong precision
- Knowledge-based errors - we don't know what we're doing
  - Fail to understand how C pointers work
  - Fail to understand the distinction between references and values in a spreadsheet

Errors of Action - you know the right thing to do but, for some reason, you don't do it

Errors of Intention - you choose the wrong course of action in response to some problem to be solved

People are subject to both types of error.

Error detection and recovery

- Slips & lapses easiest to identify and recover from as users can recognise that they have made an error
  - Provide an indication to users showing what has happened
  - Provide a mechanism to allow users to 'undo' their action
- Recovering from rule-based errors means that the system has to understand the process and rules associated with some specific intention
  - Ask users to explicitly identify what they are doing so that the rule set can be chosen
  - Associate constraints (drawn from rules) with system objects and check these after user actions
  - Automate certain processes (e.g. via wizards) so that procedural steps can't be missed out
- Recovering from knowledge-based errors is very difficult as the system has to know the intentions of the user

Slips and lapses (skill based errors) can often be avoided by good equipment design. For example, if it is dangerous to connect equipment A to equipment B then, rather than tell people this is dangerous and hope they won't make mistakes, you should design the connectors of A and B so that they are incompatible.

User interface design

- System user interfaces should be designed to reduce active errors
- Human cognitive capabilities must be considered in the user interface design
  - People are not good at repetitive work without making errors
  - People are not good at monitoring but are good at reacting
  - People's behaviour is not consistent - they may respond to the same stimulus in different ways at different times
  - People rarely follow processes exactly as specified but change them to suit local circumstances and expertise

Too much attention is paid in user interface design to how the design looks rather than whether it helps the people who are using the system.

UI design

- Error avoidance
  - Chose entry from a menu rather than input text string
  - At any point in an interaction, only allow users to make valid choices
- Error detection
  - Identify destructive actions and request user confirmation
  - Define constraints on inputs and check inputs against these constraints
  - Match inputs against allowed inputs and provide some error correction
- Error recovery
  - Provide a visual confirmation to users of what they have done
  - Provide an ‘undo’ capability

Process design

- Awareness - people in the process should be aware of the work of others
  - Public display of work
  - Indication of who is doing what
- Diversity - the process should not mandate a single way to carry out process steps
  - If different people do the same thing in different ways, an individual misunderstanding is more likely to be discovered
- Flexibility - you should avoid time dependencies in processes wherever possible
- Traceability - it should be possible to discover what has been done
  - If a problem occurs, the root of the problem can be discovered
- Team ownership - parts of the process should not be personal
  - Team ownership encourages cross-checking of work and hence error detection

Awareness is an essential feature of reliable processes. It means that people can see what each other are doing, can find and fix problems and can take over if workload becomes too high.

Organisational culture

- Many organisations take a personal view of human error and have a 'blame culture'. This looks for the person who was the source of the active error and holds them responsible
- This approach leads to:
  - A focus on concealing the error rather than fixing the problem or compensating the victim
  - A lack of problem reporting in the event of an active error so it is harder to discover latent problems in the system
  - Staff motivated to look after themselves rather than protecting the interests of the organisation

A bad example of organisational culture for safety is that prevalent in the National Health Service. Here, there is usually an attempt to ascribe blame after an accident so patients injured by medical mistakes may wait many years for compensation. There are no effective accident reporting systems, a culture of keeping quiet about errors within a professional group, a very hierarchical and status-driven staffing structure, disagreement between management and clinical staff.

High reliability organisations

- Have a collective pre-occupation with the possibility of failure and a determination to reduce the number of failures
- Have communication mechanisms in place to inform people of failures, causes and consequences
- Expect to make errors and train people to recognise and recover from these errors
- Avoid hierarchic decision making in critical circumstances
- Rehearse familiar scenarios of failure and develop novel scenarios of failure as a learning mechanism
- Focus on fixing problems not finding out who was responsible

A very good example of a high reliability organisation is the UK Air Traffic Control System. It has mechanisms for reporting and fixing errors, public displays of information, constant checking of each other's actions, extensive training, etc.

Key points

- Systems should be designed to take human fallibility into account
- Accidents occur when vulnerabilities in different layers of a system coincide like holes in a Swiss cheese
- Active failures are individual errors. Latent conditions are fundamental vulnerabilities in the system
- User interfaces should be designed for error avoidance, detection and recovery
- High-reliability organisations focus on developing a culture of avoiding failure. They learn from failures rather than try to attribute blame to individuals