ARTICLE:

The Post Office Horizon IT scandal and the presumption of the dependability of computer evidence

By **James Christie**

The Post Office Horizon scandal attracted my interest because of my extensive experience in IT audit and software testing. I have worked on fraud investigations, and also on developing and testing complex financial systems. The Horizon case covered many issues of which I have experience. I was dismayed at the Post Office's poor control over their systems and their unprofessional conduct in investigations. However, the presumption of computer dependability, which both they, and the courts, relied upon to secure convictions, truly shocked me. All of my experience has taught me that this presumption is naïve and unjustifiable. This is a personal response to the Post Office Horizon scandal.

Introduction

For the last few years, I have followed the controversy surrounding the Post Office's accounting system, Horizon. This system controls the accounts of some 11,500 Post Office branches around the UK. There was a series of alleged frauds by sub-postmasters and sub-postmistresses, many of whom protested their innocence. Nevertheless, the Post Office prosecuted these cases aggressively, pushing some of the supposed perpetrators into financial ruin, and even suicide. A number of sub-postmasters and sub-postmistresses who were affected formed the Justice for Subpostmasters Alliance (JFSA),[1] and eventually took a civil action against the Post Office, claiming (in essence) that no fraud had taken place, but rather discrepancies arose from system errors or possibly by third parties who had remote access to systems.

I was not surprised to see that the JFSA won their case in December 2019,[2] with the judge providing some scathing criticism of both the Post Office, and Fujitsu, the IT supplier. The Post Office agreed to an out-of-court settlement with the JFSA, paying £57.75 million to settle the case. Further, in March 2020, the Criminal Cases Review Commission decided to refer the convictions of a number of sub-postmasters and sub-postmistresses to the Court of Appeal, based on the argument that their prosecution involved an 'abuse of process'.[3] I will return to the prosecution tactics later.

The scandal intrigued me because of my professional experience. I have no knowledge or experience of the law, beyond a few university courses in commercial law many years ago. However, after my initial education in accountancy, I acquired wide ranging experience in IT, which was relevant to the problems of Horizon. I have worked as a system developer and designer, business analyst, project manager, IT auditor, information security manager and software testing manager and consultant. My audit experience included many technical fraud investigations and system reviews, and my development and testing experience taught me much about software quality.

The Horizon case had many features that would have caused me great concern if I had been working at the Post Office. In particular, a recurring theme is the

[1] https://www.jfsa.org.uk/.

[2] *Bates v Post Office Ltd (No 6: Horizon Issues) Rev 1* [2019] EWHC 3408 (QB).

[3] 'The CCRC refers eight more Post Office cases for appeal – bringing total to 47 so far', 3 June 2020, https://ccrc.gov.uk/the-ccrc-refers-eight-more-post-office-cases-for-appeal-bringing-total-to-47-so-far/; 'CCRC to refer 39 Post Office cases on abuse of process argument', 26 March 2020, https://ccrc.gov.uk/ccrc-to-refer-39-post-office-cases-on-abuse-of-process-argument/ ; The Criminal Cases Review Commission's process for review of convictions relating to the Post Office and Horizon accounting system (Number 2020-0040, 3 March 2020), House of Commons Library, https://commonslibrary.parliament.uk/research-briefings/cdp-2020-0040/ , https://ccrc.gov.uk/ccrc-to-refer-39-post-office-cases-on-abuse-of-process-argument/.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 4.0 International License

implications of the 1997 Law Commission recommendation[4] that in England and Wales there should be a common law presumption that computers operate correctly unless there is evidence of a problem. This presumption is the current legal position in England and Wales.[5] It is absurd. The question of reliability is essentially one of control, and we can never be certain that we are in full control of complex software. Clearly, the Post Office was not in control of Horizon in the way that I would have expected, if I had been auditing the system. I will return repeatedly to this and argue that what matters when developing and running such complex systems, is that we take, and can demonstrate that we have taken, every reasonable step to retain control.

System error and accuracy

Auditing computer systems

When I first heard about the Horizon case, I noticed the arguments were about whether the problems were caused by fraud, system error, or user error. As an auditor who has worked on the technical side of many fraud cases, the possibility that an organization might be unable to distinguish system errors from fraud makes me very uncomfortable. The system design should incorporate whatever controls are necessary to ensure such confusion cannot arise. These should include system checks to detect and either prevent or report on suspicious behaviour or mistakes, appropriate logging and retention of evidence showing which users took particular actions. Skilled software testers should also attempt to discover how the system behaves if it is used and abused by dishonest or careless users, and not simply seek confirmation that the system does what the designers expected – a serious failing, as I shall explain later.

I was an IT auditor at one of the UK's biggest insurers, the largest for personal lines business, i.e. home and motor insurance. The company only recruited people for that role who had significant IT experience. They did not believe that auditors without a technical background could cope with the demands of the job.

A routine part of our job as IT auditors was to audit live computer systems. These audits required us to establish what must happen and what must not happen, what the system must do and what it must never do.

We would ask how users, managers and developers had the ability to know that the system would do the right things, and never do the wrong things. We asked what controls were in place to prevent or detect error, loss, or fraud. We would explore progressively deeper levels of detail and work out for ourselves what controls were required. When deciding what the controls were, or ought to be, we would run through the requirement that the processing had to be accurate, complete, authorised, and timely.

'Accurate' might seem obvious, though accuracy is a far more nuanced concept than simply being right or wrong. I will return to that. 'Complete' means that everything that should be processed is processed, with no data being lost; accidentally dropping data is surprisingly easy in a complicated program. 'Authorised' meant that the output must have been created or approved by people with the appropriate level of authority, bearing in mind the necessary segregation of duties to reduce the risk of fraud. 'Timely' means that the output must not only arrive in the right place, but at the right time. Everything an insurance company does is tied in with date processing, but there is a further issue. Theft or loss might entail data moving between different time windows in a way that bypasses controls intended to prevent or detect fraud or loss.[6]

We then tested the system by looking for evidence that these controls were present and effective. We

[4] 'Evidence in Criminal Proceedings: Hearsay and Related Topics'. The Law Commission (1997). http://www.lawcom.gov.uk/app/uploads/2015/03/lc245_Legislating_the_Criminal_Code_Evidence_in_Criminal_Proceedings.pdf .

[5] Ladkin, PB. Littlewood, B. Thimbleby, H. Thomas, M. 'The Law Commission presumption concerning the dependability of computer evidence', Digital Evidence and Electronic Signature Law Review, 17 (2020), 1-14.

[6] Control reports are commonly used to help detect fraud, or anomalous behaviour that requires investigation. These often record events that have taken place in a certain time period, e.g. a working day, a shift, 24 hours, or a week. Such reports can be developed as an afterthought, once the operational system has been built. There is a danger that the criteria for the report are not aligned with the system's inner workings. It might be possible for a user to assign a fraudulent transaction to a period for which a control report has already been run, thus ensuring the transaction will not appear in a report. I know of a large fraud at a rival insurer which exploited such a control weakness. It was one of our standard IT audit tests to try to circumvent control reports.

would try to break the system, evading the controls we knew should be there, and try to exploit missing or ineffective controls. If we succeeded, we would expect, at the least, the system to hold unambiguous evidence about what we had done, when we had done it, and which user account had been used.

User error and system 'robustness'

It is inevitable that users will make mistakes and systems should be designed to allow for that. 'User error' is an inadequate explanation for things going wrong. If the system cannot cope with mistakes by users, then this comprises a system failure. Mr Justice Fraser, the judge in the Horizon case, took the same line. He expected the system 'to prevent, detect, identify, report or reduce the risk' of error, including user error. He concluded that controls had been put in place, but they had failed, and that Fujitsu had 'inexplicably' chosen to treat one particularly bad example of system error as being the fault of a user.[7] The explanation for Fujitsu's apparently inexplicable decision might lie in the legal arguments surrounding the claim by the Post Office and Fujitsu that Horizon was 'robust'. The rival parties could not agree even on the definition of 'robust' in this context, never mind whether the system truly was robust.

Nobody believed that 'robust' meant error free. That would be absurd. No system is perfect, and it was revealed that Horizon had a large and persistent number of bugs, some serious. The sub-postmasters' counsel and IT expert argued that the system could be considered robust only if it was extremely unlikely that it caused the branch losses that the Post Office had classed as frauds. They argued that the errors admitted by the Post Office would have been enough to produce these losses, and therefore the system could not be robust. The Post Office confused matters by adopting different definitions at different times, which was made clear when they were asked to clarify the point and they provided an IT industry definition of robustness that sat uneasily with their earlier arguments.

The Post Office approach was essentially top down. Horizon was robust because it could handle any risks that threatened its ability to perform its overall business role. The Post Office then took an enormous illogical leap to claim that because Horizon was robust by that definition, it could not be responsible for serious errors at the level of individual branch accounts.

Revealingly, the Post Office and Fujitsu named bugs using the branch where they had first occurred. Two of the most significant were the Dalmellington Bug, discovered at a branch in Ayrshire, and the Callendar Square Bug, also from a Scottish branch, in Falkirk. This naming habit linked bugs to users, not the system.

The Dalmellington Bug entailed a user repeatedly hitting a key when the system froze as she was trying to acknowledge receipt of a consignment of £8,000 in cash. Unknown to her, each time she struck the key she accepted responsibility for a further £8,000. The bug created a discrepancy of £24,000 for which she was held responsible.

Similarly, the Callendar Square Bug generated spurious, duplicate financial transactions for which the user was held responsible, even though this was clearly an application problem concerning the database.

The Horizon system processed millions of transactions a day and did so with near 100 per cent accuracy. The Post Office's IT expert therefore tried to persuade the judge that the odds were two in a million that any particular error could be attributable to the system. Unsurprisingly, the judge rejected this argument:[8]

> '825. Consider a hypothetical bug, bug X. Also consider that bug X impacts upon branch accounts in a single branch upon a single occasion leading to a shortfall in the branch for that branch trading period… Analysis and resolution of the correct and true situation of the branch accounts between the Post Office and the SPM for the trading period in question does not depend upon whether, in all the other millions of branch accounts, there was no such incidence of bug X… Expert IT evidence of most assistance in that exercise would be whether or not bug X exists or existed, and what were its effects. It is of no assistance to have an exercise that in effect says the statistical likelihood of any bug having an impact upon the branch accounts of that branch in that period is very low.

[7] *Bates v Post Office Ltd (No 6: Horizon Issues) Rev 1* [2019] EWHC 3408 (QB), [988]-[989].

[8] *Bates v Post Office Ltd (No 6: Horizon Issues) Rev 1* [2019] EWHC 3408 (QB), [822], [825]-[826], [879].

> 826. The section 8 analysis is, in my judgment, so riddled with plainly insupportable assumptions as to make it of no evidential value. It is the mathematical or arithmetic equivalent of stating that, given there are 3 million sets of branch accounts, and given there are so many sets of branch accounts of which no complaint is made, the Horizon system is mostly right, most of the time.'

If only 0.0002 per cent of Horizon transactions were to go wrong, then a typical day's processing of eight million transactions would lead to sixteen errors. It would be innumerate to look at one of those outcomes and argue that there was a two in a million chance of it being a system error. That probability would make sense only if one of the eight million daily transactions were chosen at random. The supposed probability is irrelevant if you have chosen a case for investigation because you know it has a problem.[9]

By their nature, legal cases are likely to involve extreme events, which are few, rather than routine, predictable ones, of which there are vastly more. A particular software failure might be exceedingly unlikely, but nevertheless occur occasionally and inevitably amongst the millions of successful transactions. Then it might well create a serious and damaging outcome, so it is highly misleading to argue that it is exceedingly unlikely that such a failure could be responsible for that outcome. To argue, as the Post Office did, that there was a probability of only two in a million that the system had caused a loss, was effectively to assume that the loss must have been caused by fraud, not system failure. It was to assume the case had been proven and try to pass off that 'proof' as evidence.

This should all be familiar to IT auditors, but it is also very basic knowledge for those working in software testing. Testers refer to these unlikely, extreme events as 'edge' cases. Any particular edge case might be extremely rare, but they come in endless varieties and they are a threat that cannot be ignored. Edge cases can kill you. This might be a metaphor in most cases, but it is literally true with safety critical systems. That is why so much testing and technical audit work focuses on the unlikely.

I, therefore, find it very strange that the Post Office persisted with its flawed perspective. A large corporation like this should have had IT auditors and software testers familiar with these concepts and problems, and who could have, and who should have, spoken out against the stance the Post Office was taking.

How accurate should accurate be?

A further concern I have about the Post Office is its attitude to accuracy. I knew all too well from my own experience of IT audit, software development and testing, that different systems in different contexts demanded different approaches to accuracy. For financial analysis and modelling it was counter-productive to try to achieve 100 per cent accuracy. It would be too difficult and time consuming. This pursuit might introduce such complexity and fragility to the system that it would fail to produce anything worthwhile, certainly in the timescales required. A system that is 98 per cent accurate might be good enough to provide valuable answers to management and quickly enough for them to exploit them. Even 95 per cent could be good enough in some cases.

In deeply complex systems it might not even be possible to say what 100 per cent accuracy means, or to know whether it has been achieved. Systems can run quite satisfactorily with levels of inaccuracy that are considered acceptable in context. With complex financial applications an honest and constructive answer to the question 'is the application correct?' would be some variant on 'what do you mean by correct?', or 'I don't know – it depends'. It might be possible to say the application is definitely not correct, if it is producing obvious nonsense. But the real difficulty is distinguishing between the seriously inaccurate, but plausible, and the acceptably inaccurate that is good enough to be useful. Often accuracy is a question of experienced judgment rather than arithmetic.

In other contexts, when dealing with financial transactions and customers' insurance policies, we really do need a far higher level of accuracy. Arriving

[9] The Post Office's expert came very close to committing the Prosecutor's Fallacy. This assumes that the probability of incriminating evidence being linked to a randomly selected person is equal to the probability that the defendant is innocent. The probability of a certain action occurring randomly and leading to a specific outcome is not the same as the probability that the action was the explanation if the outcome has already happened and been detected. https://en.wikipedia.org/wiki/Prosecutor%27s_fallacy.

at the correct premium to charge a policyholder requires precise and complicated calculations based on numerous factors. Data migrations from old to new systems are common in insurance IT and these are surprisingly complex exercises, which are prone to serious error. Customer policies can easily be dropped during a migration. If we do not reach 100 per cent, we need some way of spotting and handling the exceptions. These must not be dismissed as remote theoretical risks that we can live with. They are people's insurance policies or claims payments. Arguing that losing a tiny fraction of 1 per cent is acceptable, would have been appallingly irresponsible, and I cannot stress this enough; as IT auditors we would have come down hard, very hard, on anyone who tried to take that line.

After writing the first version of this article, I was fascinated to read Peter Bernard Ladkin's article 'Robustness of software'.[10] He made essentially the same point, that acceptable levels of accuracy in computer modelling systems are very different from those required in a transaction processing system. I was encouraged to see an academic writer appreciating the significance of the conclusion I had formed as a practitioner.

Perhaps the most famous quote in software testing comes from Gerald M. Weinberg, who shaped much of that community's thinking. 'Quality is value to some person'.[11] When forming an opinion about the quality of a system, we have to be clear about its role and what the main users need. Horizon was performing different roles for different people, which is hardly unusual, but in that case, it led to persistent confusion about the quality of the system. Worse, the Post Office actively spread and encouraged confusion about quality. It appears that Horizon allowed the Post Office to manage its corporate accounts to an acceptable level of accuracy for the purposes of the whole organisation. However, the system was also used for the day-to-day management of sub-postmasters' businesses. For this purpose, for this community, it appears that the level of quality was abysmal.

There are some things a system should always do, and some it should never do. Systems should never lose people's data. They should never inadvertently produce apparently fraudulent transactions that could destroy small businesses and leave the owners destitute. The amounts at stake in each individual Horizon case were trivial as far as the Post Office was concerned, immaterial in accountancy terminology. But for individual sub-postmasters and sub-postmistresses they were big enough to change, and to ruin, lives.

The willingness of the Post Office and Fujitsu to absolve the system of blame and accuse users instead was such a constant theme that it produced a three-letter acronym that I had never seen before: UEB, or user error bias. Naturally, this arose on the claimants' side. The Post Office never accepted its validity, but it permeated their whole approach, as if to say: 'Horizon was robust from the perspective of the senior executives, therefore any discrepancies must be the fault of users, whether dishonestly or accidentally, and we could proceed safely on that basis'. I knew from my experience that this was a dreadful state of mind with which to approach fraud investigations.

Fraud and evidence

Fraud investigations

Although I worked on many fraud cases that resulted in people going to prison, I was never required to appear in court to give evidence. This was because we built our case so meticulously, with an overwhelmingly compelling set of evidence, that the fraudsters always pleaded guilty, rather than risk antagonising the court with a wholly unconvincing plea of innocence.

We always had to be aware of the need to find out what had happened, rather than simply sift for evidence that supported our working hypothesis. We had to follow the trail of evidence, and remain constantly alert to the possibility we might miss vital, alternative routes that could lead to a different conclusion. It is very easy to fall quickly into the state of mind that the suspect is definitely guilty and ignore anything inconsistent with that belief. Working on these investigations gave me great sympathy for the police carrying out detective work. If you want to make any progress you cannot follow up everything,

[10] Ladkin, PB. 'Robustness of software', Digital Evidence and Electronic Signature Law Review, 17 (2020), 15-24.
[11] See an interesting blog post by Mr Weinburg dated 23 September 2012 on this topic at http://secretsofconsulting.blogspot.com/2012/09/agile-and-definition-of-quality.html.

but you have to be aware of the significance of the choices you do or do not make.

In the cases of which I was a part, there was a clear and obvious distinction between the investigators and the prosecutors. We, the auditors, would do enough investigation to be confident we had the evidence to support a conviction. We would then present that package of evidence to the police, who were invariably happy to continue with a case where someone else had done the initial investigative work.

The police would always do some confirmatory investigation of their own. They would often gather non-computer evidence that would have been circumstantial or inconclusive on its own, but which supported the computer evidence. However, it was our work that pulled together the whole story that would secure a conviction. The prosecution of the cases was the responsibility of the Crown Prosecution Service in England and Wales, and the Procurator Fiscal Service in Scotland. That separation of responsibilities helps to guard against some of the dangers that concerned me about bias during the investigation.

In England and Wales, this separation did not apply to the Post Office, which for historical reasons employs its own prosecutors. It also has its own investigation service. There is nothing unusual about internal investigators, but when they are working with an in-house prosecution service, that creates the danger of encouraging unethical behaviour. In the case of the Post Office, the prosecution's case has been subject to harsh criticism.[12]

The usual practice was to charge a sub-postmaster or sub-postmistress with theft and false accounting, even if the suspect had highlighted a problem with the accounts, and there was no evidence that he or she had benefitted from a theft, or even committed one. Under pressure, sub-postmasters and sub-postmistresses would usually accept a compromise and enter a plea of guilty to false accounting, which would allow the Post Office to pursue them for the losses.

What made this practice shameful was that in some cases it appears that the Post Office should have known it had no evidence that would secure a conviction. In the earlier 'Common Issues' trial, concerning the contractual relationship imposed by the Post Office, Fraser J stated that if a sub-postmaster or sub-postmistress (SPM) notified the Post Office of a problem via the Helpline,[13] 'the issue of an SPM as an agent deliberately rendering a false account… simply does not, in my judgment, arise'. This does not seem to have troubled the Post Office's prosecutors. They were protecting the interests of the Post Office. The question is whether the end justified the means.

The argument that the prosecution tactics were deplorable is being taken very seriously. The Criminal Cases Review Commission has referred a number of Horizon cases for appeal, on the grounds of 'abuse of process' by the prosecution.

The approach taken by Post Office investigators and prosecutors was essentially to try and ignore the weakest points of their case, while concentrating on the strongest points. This strikes me as fundamentally wrong. It is unprofessional and unethical. It runs counter to my experience.

Quality of evidence

Although I was never called to appear as a witness in court, when I was assembling the evidence to be used in a fraud trial, my preparations were always based on the assumption I would have to face a barrister or advocate who had been sufficiently well briefed to concentrate on any possible areas of doubt or uncertainty. I had to be prepared to face an aggressive questioner who could understand where weak points might lie in the prosecution's case. The main areas of

[12] *Bates v Post Office Ltd (No 6: Horizon Issues) Rev 1* [2019] EWHC 3408 (QB) at [229], the judge, at sub-paragraph 6, criticized a senior Post Office employee for providing a misleading witness statement which tried to conceal that the Post Office had been warned, and had admitted, in 2013 that Horizon could automatically generate changes that looked as if they were made by a user, and that this erroneous data might be used in a prosecution. The Post Office had taken no action to correct the system six years later, yet persisted with prosecutions based solely on system evidence. The judge said: 'Finally, and in my judgment also importantly, any risks that Fujitsu and/or the Post Office" cannot clearly see what has happened on the data available to" them and "this in itself may be misinterpreted when giving evidence and using the same data for prosecutions" is a serious matter. I do not understand how a report containing such a reference to such a serious matter could be mis-summarised by Mrs Van Den Bogerd in her witness statement dealing with Mr Coyne's analysis of this.'

[13] *Bates and Others v Post Office Limited (No 3)* [2019] EWHC 606 [819]-[821].

concern were where it was possible that data might have been erroneous, that it might have been tampered with, or where it was possible that someone else had taken the actions that we claimed were the actions of the accused. Our stance was that our case was only as strong as the weakest link in the chain of evidence. I believed I had to be ready to explain why the jury should be confident 'beyond reasonable doubt' that the accused was guilty.

As an IT auditor I worked at the head office of a company based in Scotland, but whose remit for Internal Audit extended throughout the UK. We wasted no time agonizing over the implications of the differing treatments that English and Scots Law applied to computer evidence. The niceties of hearsay were a mystery. Corroboration mattered, and we were keen to hear what the police could find to back up our work. The Scottish courts require such corroboration, but as it happened, I worked on only one Scottish case, and the fraudster confessed immediately when she was confronted with the computer evidence.

It might seem that our ignorance and even indifference to the legal status of computer evidence was cavalier. I would argue that it was quite the reverse. We were experts in IT and audit and had to cover a vast range of subject matter. Attempting to keep up with changes in the law as well would have been a distraction. We applied the highest possible standard to our work and our evidence, rather than looking to the law to see what we could get away with. We left the law to the lawyers.

At that stage in my career I had worked only in software development and IT audit. My experience as a software testing manager came later. However, I was already sufficiently experienced to know how fallible software is. The idea that we could simply assert the validity of a printout because it came from a computer would have been embarrassing. I would have refused to do it, although it is inconceivable that my management would have put me in that position. That, however, was the clear implication of the Post Office's argument when prosecuting Horizon cases, and of the 1997 Law Commission recommendation that in England and Wales there should be a common law presumption that computers operate correctly unless there is evidence of a problem.

Also, thinking our way through the possible weaknesses in our case was an intrinsic part of our normal work. My preparation for a difficult cross-examination was more of a thought experiment than a practical rehearsal. We monitored the company's IT activities to provide reassurance that they were properly controlled and that risks were managed. If our evidence lacked credibility, it would mean that there were weaknesses in systems or the management of IT that had to be tackled, regardless of any possible legal proceedings. The ability to provide compelling evidence was simply one facet of running a well-managed IT installation.

It was theoretically possible that a systems programmer (i.e. one of the technical experts who manage the operating system as opposed to the business applications) could have bypassed access controls and tampered with the logs, but it was utterly implausible that they could have set up a web of consistent evidence covering many applications over many months and years, and that they could have done so without leaving any trace.

In any case, these systems programmers lacked the depth of application knowledge required to undertake such activities. Some applications developers, and the IT auditors, did have the application knowledge, but they lacked the necessary privileges to subvert access controls before tampering with evidence.

It is worth stressing that our fraud programs were not normal application programs. They were written by the IT auditors for each particular case, trawling through the vast historical archive, matching insurance policies, claims, cheque payments, names, addresses, log-in accounts and establishing the patterns that indicated a fraud. The nature of our approach meant that any errors in the source data would have excluded that data from our reports. It was unlikely that random errors could have assigned to data characteristics that allowed them to be picked up by our programs, given they were looking for consistent patterns across applications. Any errors would have hidden the affected records from our programs by destroying the link to the pattern for which we were searching.

In addition to the listings with the incriminating evidence about the fraud our packages of evidence included the source code, Job Control Language (JCL) checks for all the fraud detection programs and run logs. These would all have been available to the defence so that an expert witness could dissect them. We not only had to do the job properly, we had to be confident we could justify our work in court.

JCL[14] is an IBM mainframe computer language that instructs the operating system on how to run an application. The JCL specifies which programs are run, using which files and storage devices for input or output, and what should be done if a problem is encountered. The JCL and run logs together provide a link between the raw data and the fraud reports. They provided evidence about when the jobs had been run, and also a full audit trail from the output data back through the logic to the input data. An independent examiner could therefore have come in, looked at our work, and recreated our final reports with their own programs because we had documented the state of every input at the time our reports were run.

Another theoretical possibility was that a different employee had logged into the accused's account to make fraudulent transactions. However, we could match these transactions against network logs to show that the actions had always been taken from the terminal located on the accused's desk during normal office hours on days when they had registered their presence in the office. I could sit at my desk in head office and use a network monitoring tool to watch what a suspect was doing hundreds of miles away. In one case I heard a colleague mention that the police were trailing a suspect around Liverpool that afternoon. I told my colleague to telephone back and tell the police they were following the wrong man. Our suspect was sitting at his desk in Preston, and I could see him working. Half an hour later the police telephoned back to say we were right.

In any case, fanciful speculation that our evidence had been manufactured can be offset by evidence of motive – for instance, where the accused was enjoying a lifestyle well beyond his or her salary, whereas those who might have tampered with evidence had nothing to gain and a secure job, pension and mortgage to lose.

I have tried to explain how we approached such tasks and the thought processes we followed so that it is possible to understand why I was shocked to read about what happened at the Post Office. We knew that systems were fallible, and that any presumption that computers were always correct was nonsense. That awareness was partly why we investigated and prepared meticulously in case we had to appear in court, but this was also the standard that was expected of all IT work. That level of professional preparation goes a long way to explaining why we were never called to give evidence. The fraudsters always admitted the crime when they realised how strong the evidence was.

Superusers going 'off piste'

One of the most contentious aspects of the Horizon case was the prevalence of Transaction Corrections, that is, corrections applied centrally by IT support staff to correct errors. The Post Office seems to have regarded these as being a routine part of the system, in the wider sense of the word 'system'. But it regarded them as being outside the scope of the technical Horizon system. They were just a routine, administrative matter.

I came across an astonishing phrase in the judgment[15] lifted from an internal Post Office document. 'When we go off piste we use APPSUP'. That is a powerful user privilege that allows users to do virtually anything. It was intended 'for unenvisaged ad-hoc live amendment' of data. It had been used on average about once a day and was assigned on a permanent basis to the user accounts of all the IT support staff looking after Horizon.

That is a shocking state of affairs. Where I worked as an IT auditor, nobody was allowed to have an account with which they could create, amend or delete production data. There were elaborate controls applied whenever an ad hoc or emergency change had to be made.

The change would be made as a production batch job. It would be set up by a developer and passed to Operations to check and apply, after approval by the developer's manager. Emergency changes were sometimes required overnight. The developer would take control of a superuser[16] account, by signing a register and being handed a sealed envelope with the password. All actions taken with that account were logged and inspected. The developer was accountable for everything done with the account until they handed back control to Operations, who would change the password, placing the new one in an envelope with the responsible person signing across

[14] Job Control Language.

[15] *Bates v Post Office Ltd (No 6: Horizon Issues) Rev 1* [2019] EWHC 3408 (QB), [390]-[394].

[16] https://en.wikipedia.org/wiki/Superuser.

the seal. The incident and the change would be independently checked the next working day.

Audit would carry out periodic spot checks on the register, envelope and logs to establish that the password had been changed after the last time the account had been active. It was a tiresome, bureaucratic process, but that was the point. It was meant to provide essential emergency access in a controlled way. It was not meant to be convenient.[17]

If I failed to discover that powerful user privileges were used in an uncontrolled way on the first day of a two-day, high level, IT installation audit I would have been embarrassed. It is that basic. However, the Post Office's internal auditors do not have the excuse of incompetence. The external auditors, Ernst & Young, raised the problem in 2011. It is inconceivable that internal audit was unaware of a problem identified by the external auditors. It is a significant decision for the external auditors to raise a control weakness in their audit report, and it should be highly embarrassing for the internal auditors. Once the external auditors have taken that step, they are duty bound to follow it up, and to keep doing so. Even if internal audit had wanted to ignore the problem, they could not have done so. There is no excuse for inaction, or tardiness.

That the serious underlying problem was not tackled quickly, if at all, is revealed a few paragraphs later in the Horizon judgment.[18] The Chief Systems Architect for Fujitsu (the outsourcing IT services supplier) on the Post Office account conceded that before 2015 the only control over the use that was made of the highest level of access privilege by database administrators, was a record of when they logged on and off. These people could insert, amend and delete branch data, and do so without any control.

I am not sure if the reader will realise how startling the phrase 'off piste' is in this context to someone with solid IT audit experience in a respectable financial services company. Picture the reaction of a conventional auditor on discovering that a company's petty cash fund was held in a bucket with a supply of blank, signed cheques at reception, for staff convenience if they have to dash out to purchase emergency supplies to keep the company going. It is not just a question of users holding a superuser privilege all the time, bad though that is. It reveals a great deal about the organisation and its systems if staff members have to change live data routinely. An IT installation that cannot control superusers effectively probably does not control much and would be badly lacking in credibility if it contested legal proceedings that hung on its computer evidence and the opposing barrister understood the implications. Perhaps the repeated, successful Horizon prosecutions show that the legal process is not as stringent as it should be.

We had to be confident in the integrity of our data. If we had discovered a member of staff with permanent update access to live data, for every occasion when they went 'off piste', we would have raised the problem with IT management as a matter of extreme urgency, and escalated our concern as far as necessary. We would not have ceased pressing for an effective solution until the matter was fully resolved. If the company had been facing legal proceedings that centred on how confident we could be in our systems and data, we would have argued strongly that the organization should settle once we were aware of the 'off piste' problem.

The only times when work has ever affected my sleep have been when I knew that the police were going to launch dawn raids on suspects' houses. I would lie in bed thinking about the quality of the evidence I had gathered. Had I got it all? Had I missed anything? Could I rely on the data and the systems? I worried because I knew that people were going to have the police hammering on their front doors at 6 o'clock in the morning.

I am appalled that Post Office investigators and prosecutors appeared to approach fraud investigations with the attitude 'what can we do to get a conviction?'. They pursued the sub-postmasters aggressively, being fully aware of the system flaws and the control weaknesses in Horizon and the Post Office.

[17] These emergency access procedures are often referred to as 'break glass' processes, referring to the common signs 'in case of emergency break glass', which makes it clear that these accesses are not routine actions.

[18] *Bates v Post Office Ltd (No 6: Horizon Issues) Rev 1* [2019] EWHC 3408 (QB), [395]-[398].

The presumption that computers work correctly

A difficult presumption for IT experts to take seriously

Throughout the series of prosecutions of sub-postmasters and subpostmistresses, and the Horizon case itself, the Post Office relied heavily on the presumption in England and Wales that computers operate correctly unless there is evidence of a problem.[19]

> ‘We provisionally proposed that section 69 of PACE be repealed without replacement. Without section 69, a common law presumption comes into play:
>
> ‘In the absence of evidence to the contrary, the courts will presume that mechanical instruments were in order at the material time.’
>
> Where a party sought to rely on the presumption, it would not need to lead evidence that the computer was working properly on the occasion in question unless there was evidence that it may not have been…’

For an experienced IT practitioner this is a frustrating presumption to tackle. It is so deeply flawed, for many reasons, on various levels, that if most practitioners were faced with such a claim in an informal social setting, the temptation would be to snort derisively and laugh,[20] rather than take it seriously.[21] Nevertheless, when such a presumption guides the law of the land, sceptics from the IT community have a responsibility to engage constructively and attempt to explain why the presumption is out of touch with the reality of computer systems.

The strongest argument for the presumption’s retention is that it is a pragmatic response to the problems entailed in proving the reliability of a complex computer system.[22] The presumption, therefore, is a necessary trade-off, rather than being a naïve expectation of perfection; it balances the difficulties of making categorical statements about system behaviour against the need for courts to operate effectively. It is an evidential presumption that can be rebutted rather than a rigid rule. However, this presumes that the party producing the evidence provides full disclosure of information that would allow the other party to challenge the evidence. It also presumes that the other party has the financial and technical resources to mount a successful challenge. In the Horizon prosecutions the defendants were unable to overcome that hurdle, and, in effect, the presumption carried far more weight than it would merit as an evidential presumption.

My experience has persuaded me that while it is certainly extremely problematic to prove, or make categorical statements about, system reliability, the contradictions and problems raised by the presumption make it unsustainable.

What does the presumption apply to?

My first concern is that the presumption cannot credibly be applied to all software. If we are considering the validity of evidence, there is a difference in principle between fraud detection programs and operational systems. The Law Commission, and judges and lawyers who have relied on the presumption, seem to assume that computer evidence has been produced as part of a dispassionate, formal system. Printouts may have been created by investigators with the explicit intention of proving the case being argued in legal proceedings. It is surely, therefore, quite unreasonable to presume that these artefacts created by humans using computers as a tool, should be correct when the case being argued by the party that provided the evidence is essentially the same: that

[19] ‘Evidence in Criminal Proceedings: Hearsay and Related Topics’. The Law Commission (1997), 13.13-13.14. http://www.lawcom.gov.uk/app/uploads/2015/03/lc245_Legislating_the_Criminal_Code_Evidence_in_Criminal_Proceedings.pdf.

[20] See Appendix for an informal Twitter poll of IT practitioners. The poll is hardly scientific, but it does illustrate the point.

[21] I understand that the presumption may be weaker than it is made out to be – in that it only operates as an evidential presumption, but that does not seem to be any help to the person to whom it is directed at, for which see the article by Paul Marshall, ‘The harm that judges do – misunderstanding computer evidence: Mr Castleton’s story’, 17 (2020) Digital Evidence and Electronic Signature Law Review, 25-48.

[22] See the comment by Peter Sommer following this article by Steven J Murdoch. https://www.benthamsgaze.org/2019/12/19/resolving-disputes-through-computer-evidence-lessons-from-the-post-office-trial/#comment-136259.

printouts tell an irrefutable story. It would amount to presuming the guilt of the accused and offering that presumption as evidence.

This might imply that a line should be drawn between operational and investigatory, or control, software. But where do you draw the line? As an auditor I have asked for controls to be embedded within operational software to identify problems for investigation. Exception reporting is a routine function of systems. Once you acknowledge that a line should be drawn, the justification for the presumption as a general proposition starts to crumble. The presumption is inconsistent with the way that software systems work and is especially incompatible with the nature of the complex systems that feature in disputes. That takes me on to my second objection to the presumption.

The presumption confuses the computer with the software, that is the machine with the human attempts to do something useful with the machine. Computer hardware is, generally, reliable today. The operating systems that run computers are also dependable, if not quite to the same extent as the hardware. However, application software is notoriously unreliable. That is not a criticism of developers. It is the nature of software.

Computer systems exist on multiple levels, and for a system developer the distinction between software and hardware becomes blurred. The detail of the infrastructure becomes largely opaque as successive layers interpret the commands of higher layers and manage lower ones. Without such abstraction, which hide the details of how devices are controlled and allow developers to work in languages comprehensible to humans, it would be impossible to build anything worthwhile in acceptable timescales. But this abstraction distances the fallible human from the reliable machine and creates endless scope for error. In particular, web applications are built on layers of abstraction that can disguise the reality that many components are not even notionally under the control of the organisations that build and own software.

Whether 'in order' and 'working correctly' are meaningful

Software has always been unreliable, but the way it is built today, and the way that systems develop and evolve, has created a level of complexity that make it impossible for responsible experts to say whether large computer systems are 'in order'. Over the last decade this has prompted increasing debate and research within IT, particularly among those working with safety critical systems where errors will kill people. This community has had to be realistic about the nature of software and computer systems. It has much to teach the rest of the IT world, where there has traditionally been a reluctance to concede the implications of the fallible nature of their products.

I will provide an overview of how increasing complexity has undermined our confidence in what we can know and say about how systems behave. I gave a keynote talk on this subject, 'Facing the dragons – dealing with complex unknowable systems', at the EuroSTAR Software Testing Conference in The Hague in November 2018. The conference theme was 'I don't know / je ne sais pas'. I was awarded the best paper award. The paper was released as an ebook,[23] and is also available in expanded form on my blog as 'The dragons of the unknown'.[24]

My talk blended my own experiences working on complex systems with recent work in the fields of safety critical systems and resilience engineering. I was not announcing anything new or important that I had done, but drawing connections between different disciplines and explaining the relevance to software testers.

In my talk, I discussed the thinking behind the audits of live systems when I was an IT auditor. The auditees usually assumed we would check the systems against the original specifications for requirements and design. We never looked at them. They were irrelevant. The specifications were a partial and flawed picture of what was required at the time that the system was built. They were not necessarily relevant to the business risks and problems facing the company at the time of the audit, so we could not use them to determine whether the system was 'correct'.

[23] Christie, J. 'Facing the dragons – dealing with complex unknowable systems'. EuroSTAR (2018). https://huddle.eurostarsoftwaretesting.com/resources/functional-testing/facing-facing-dragons-dealing-complex-unknowable-systems/.

[24] Christie, J. 'The dragons of the unknown', author's blog (2019), https://clarotesting.wordpress.com/the-dragons-of-the-unknown/.

The system's compliance, or failure to comply with the specifications told us nothing useful about what the system was doing or should be doing. We never thought it was credible that the specifications would have been updated to reflect subsequent changes, and there would inevitably have been many changes, not all of them visible to those responsible for the system documentation (see my previous point about layers of abstraction).

We were interested in the actual behaviour of the people using the system, not what the analysts and designers thought they would or should be doing. Once people start using a computer system in a complex setting, they will make the system do things that the designers never imagined. They will understand the gaps in the system – the gaps in the designers' understanding. The users then use their own ingenuity to adapt and extend the system. These user variations are usually beneficial and help the system work as the business requires. They can also be harmful and provide opportunities for fraud or accidents. This is just one of the reasons why it is vital that testers do not focus solely on confirmation that the system does what the designers expected. The testers must try to inform the key users how the system will behave in unexpected circumstances, or if it is used in unexpected ways.

These user variations are inevitable in complex systems, and I will expand on this point in the next section. The safety critical systems community likes to talk about tractable and intractable systems. They know that the complex socio-technical systems they work with are intractable, which means that they cannot even describe with confidence how they are supposed to work. The question is whether this rules out the possibility of offering a firm opinion about whether they are working as intended. Is it meaningful to talk of complex systems as being 'in order' or 'working correctly'? I believe that any use of such phrases should be heavily qualified in such a way that rules out any presumption in the reliability of computer systems. The presumption is particularly inappropriate with the complex systems that are most likely to feature in legal proceedings.

The troubling nature of complex systems

Traditionally in IT we thought that well designed and well-built systems were deterministic. A deterministic system will always produce the same output, starting from a given initial state and receiving the same input. Probabilistic, or stochastic, systems are inherently unpredictable and therefore non-deterministic. 'Stochastic' is defined by the Oxford English Dictionary as 'having a random probability distribution or pattern that may be analysed statistically but may not be predicted precisely.' The idea that system behaviour can be 'random' and 'may not be predicted precisely' is deeply troubling to IT people, but this is what we increasingly have to deal with.

I was trained in a world where we assumed that non-determinism meant a system was badly designed, inherently buggy, and untestable. In particular, software testers needed deterministic systems to do their job. Traditionally, we tested systems as if they were calculators, which should always produce the same answers from the same sequence of button presses. It was therefore the job of designers to produce systems that were deterministic, and testers would demonstrate, whether or not the systems met that benchmark. Any non-determinism meant a bug had to be removed.

As technology has changed and spread into every area of our lives, the presumption of determinism has weakened. It can apply only within carefully defined limits. You can argue, quite correctly, that a computer program is necessarily deterministic. It does what it is coded to do: outputs can always be predicted from the inputs, provided you are clever enough and you have enough time. For a single, simple program that is certainly true. It might be extremely difficult to predict the behaviour of a fearsomely complicated program, but we can respond constructively to that with careful design, and sensitivity to the needs of testing and maintenance.

However, if we draw the context wider than individual programs and look at how they combine into systems, the weaker becomes our confidence that we can know what should happen. Once we are looking at complex socio-technical systems, that is systems where people interact with complex technology, then any reasonable confidence that we can predict outcomes accurately has evaporated. These are the reasons.

Even if the system is theoretically still deterministic, we do not have sufficiently powerful brains to comprehend its behaviour, so for practical purposes the system becomes non-deterministic. When I worked on testing complex systems, we needed the computer to make any sense of what the computer itself was doing, which has its own drawbacks. The new systems were producing information that was not available by any other means. There were no oracles[25] against which we could assess the output. We wrote complex programs to test the system, but if the testing programs contradicted the system it was hard to decide which was right. If they gave the same answers, we had to consider the possibility that both were wrong, perhaps sharing the same flawed assumptions. The data were constantly changing and we could never be sure that the different output was due solely to the different data, and not to other factors. The computer could take us only so far. As I have indicated earlier, in order to assess accuracy we had to rely on experienced judgment. We were also guided by business rules, as will be explained later.

In a deeply complicated system, things will change that we are unaware of. There will always be factors about which we are ignorant, or whose impact we cannot know about. I was a test manager for Y2K, the Millennium Bug. We had to check, convert and test all the old systems that had been running quietly in the background for years, decades even. Even the newer systems used old components, some of which were older than the staff working on them, undocumented and written by people who had long since retired. I was considered an expert in both the technical and business domains, but I was acutely aware I had only a superficial understanding of many of the systems for which I was responsible. Both my employers and I were painfully aware that nobody knew more than people like me, people who had only a partial knowledge of the systems. We had lost control of the massively complex financial systems we were managing.

Media stories about Y2K assumed the main concern was that systems would crash. System crashes are hardly trivial, but for experts they are invariably a manageable problem. The concern is not whether a system can be fixed, but how quickly. Of course, if all systems had crashed simultaneously it would have been impossible to manage that problem. But for individual systems our real fear with Y2K was not that failing systems would crash, but that they would keep running, while producing inaccuracies, of which we were unaware. A failing system might appear to be working as normal, but gradually introduce serious distortions. That takes us back to my earlier comments about accuracy. How can we say a complex system is accurate if we are not confident we can detect all the inaccuracies? How can a non-expert, or even an expert, possibly presume the system is working properly?

The Y2K experience changed the way I thought about systems. We were extremely humble and modest about what we knew, but there was a significant amount we did not even know that we did not know. At the end of the lengthy, meticulous job of fixing and testing to ensure our systems could cope with the year 2000, we thought we had allowed for everything – in the high risk, date sensitive areas at least. We were amazed how many fresh problems we found when we were given the use of a dedicated mainframe partition, effectively our own mainframe computer, and ran it with future dates.

We discovered that there were vital elements (operating system utilities, old vendor tools, etc) waiting in the underlying infrastructure that did not look like they could cause a problem, but which interacted with application code in ways we had not understood. The fixed systems had run satisfactorily with overrides to the system date in test environments that were built to mirror production, but they crashed when they ran on a mainframe whose operating system was set to the future dates. We were experts, but we had not known what we did not know. Twenty years ago we had lost control and lost sight of what we were doing. Since then the situation has become only worse with the enormous increase in interlinked web applications.

Further, there are severe limits about what we can say about the whole system from our testing of the components. A complex system behaves differently from the aggregation of its components. The whole is more than the sum. Complex systems have emergent qualities. As the system is run and the components interact, the system adapts and produces unpredictable behaviour. Inevitably it is these complex systems that feature in legal proceedings, and this will increasingly be the case. Simple and trivial systems are unlikely to be a source of contention.

My IT development and testing experience has been largely with huge financial systems that evolved from being merely complicated into complex, adaptive,

[25] In software testing an oracle is an independent source of verification that can be used to decide whether a system passes a test.

unpredictable systems. When such a system is subjected to rigorous testing and a test passes, there are strict limits to what we should say with confidence about the system. The test passed now, in this setting. The system will not necessarily perform in the same way in the future. Changes will disrupt the system, even if there is nothing wrong with the change. Even carefully tested changes that fix errors can cause unexpected later problems by creating new pathways to failure.

The adaptive nature of complex systems means that each of the components, or programs, might be operating perfectly as they were designed, but the system might fail because they interact in unexpected ways. On the other hand, individual components might be flawed, but the system as a whole could be operating satisfactorily.

Experts who have worked with complex systems, understand that these systems run in a permanently flawed state, in a 'degraded mode' as Richard Cook wrote in a highly influential 1998 article.[26] Cook was obviously familiar with the realities of complex systems when he described how people keep flawed systems running, making judgments that are informed by experience, but not certainty. They are essentially calculated gambles.

I have tried to explain why we auditors never bothered to look at the original specifications for systems. Richard Cook put it simply in a 2012 talk,[27] drawing a crucial distinction between the system as imagined by their creators and the real system that could be found in live use: 'Systems as imagined are static and deterministic. Systems as found are dynamic and stochastic.'

We were not interested in the system that a designer had imagined a few years ago. We wanted to find out about the real system as it was currently being used. What we found would often shock the IT experts who supported the system and the business managers whose staff used it.

If anyone does claim they fully understand a complex socio-technical system then one of the following applies, setting aside the obvious possibility that they are lying:

1. They genuinely have no idea of the true complexity of the system. They have a superficial understanding and do not know enough to understand how little they understand.

2. They did have a good understanding of the system once upon a time, when it was simpler, before it grew and evolved into a complex monster. Their understanding relates to the time when the system was developed. There have since been changes to programs, the infrastructure or the technical or business environment that the professed experts do not know about or fully understand.

3. They understand only part of the system – probably one of the less complex parts, and they are ignoring the rest, perhaps the labyrinthine back-office processing, or awkward interfaces with other systems. They might have focused on the part that they can understand, then offer an opinion based on that. They may well understand, or have understood, the component programs without appreciating how the system as a whole behaves and adapts when these components interact.

Even if self-professed experts qualify their opinion to take account of the limits of their knowledge, other people might not appreciate the significance of that qualification. It might not be noticed that the problem has been defined so that it is comprehensible, but not realistic. That is an issue that applies especially to the distinction between the behaviour of the components and the whole system.

If in doubt, people in IT are too fond of bluffing and are happy to appear more certain than they have any right to be. Sadly, that occurs in many organisations which value confidence over realism.

These flawed, complex systems do a valuable job, but any judgment of that value depends on what is needed from them and who needs it. Systems might

[26] Cook, R. 'How complex systems fail'. Cognitive Technologies Laboratory, University of Chicago (1998). https://www.researchgate.net/publication/228797158_How_complex_systems_fail/link/5caf748a299bf120975f697e/download.

[27] Cook, R. 'How complex systems fail', (talk). Velocity Web Performance and Operations Conference (2012). https://www.youtube.com/watch?v=2S0k12uZR14 , starting at 14:48.

be adequate for some purposes and users, but not others. As I argued in my earlier comments about accuracy, inaccurate systems can run quite satisfactorily if the level of inaccuracy is acceptable to the users who matter. Horizon would make a good case study. It seems to have operated satisfactorily to provide the Post Office's corporate accounts. For some sub-postmasters and sub-postmistresses, however, it was disastrous.

Human error is the result of a problem, not the cause

One of the damaging consequences of excessive confidence in the reliability of computer systems has been a tendency to blame the problems and accidents that occur on the people operating the systems. It is a dangerous and unjust over-simplification to blame undesirable outcomes on human error simply because the user was not operating the system as expected. As I have explained, the notion of 'as expected' is difficult to sustain.

Adaptations introduced by users were usually present when the system was operating successfully, and may well have been the reason that it was previously working satisfactorily. If systems are to remain valuable, they adapt in the hands of skilled users. System behaviour changes in ways that are impossible to predict.

The expansion in system complexity and the demands that are placed on users create steadily increasing pressure on the users to do more, do it faster and do it in more efficient ways. However, introducing new controls, alerts and warnings, ostensibly to help users, increases the complexity of the technology and creates opportunities for new types of failure. Perversely, these new features can add to the burden on the people and can make the problem worse.

I strongly recommend the story told by Bob Wachter in 'The overdose: harm in a wired hospital'.[28] A patient at a hospital in California received an overdose of 38½ times the correct amount. Investigation showed that the technology was in order. All the individual systems and components performed as designed. They highlighted potential errors before they happened. So, someone obviously made a mistake; someone was careless or negligent. That would have been the traditional verdict. However, the hospital took a different approach and allowed Wachter to interview everyone involved in each of the steps. He observed how the systems were used in real conditions, not in a demonstration or test environment. During the course of five articles, he told a compelling story that will force any fair reader to admit 'yes, I would probably have made the same errors in those circumstances'.

Fortunately, the patient survived the overdose. The hospital staff involved were not disciplined and were allowed to return to work. The hospital had to think carefully about how it would try to prevent such mistakes recurring. The uncomfortable truth they had to confront was that there were no simple answers. Blaming human error would have been an evasion of the real issues. Adding more alerts would compound the problems staff were already facing; one of the causes of the mistakes was the volume of alerts swamping staff and making it hard, or impossible, to sift out the vital warnings from the important and the merely useful.

Some system changes were required and made, but the hospital realised that the greater problem was organisational and cultural. They made the brave decision to allow Wachter to publicise his investigation, and his series of articles is well worth reading.

One of the hard lessons illustrated by that case was that focusing on making individual components more reliable had harmed the overall system. All the components were safe, or as safe as anyone could judge from examining and testing them in isolation, but the overall system almost killed a child. The story is an important illustration of the maxim in the safety critical community that trying to make systems safer can make them less safe. Trying to automate people out of complex systems has the perverse effect of making the remaining work that must be performed by humans more demanding and error prone. Trying to make these systems easier to use creates fresh demands on the people as the improvements are always seized upon to demand greater efficiency and productivity.[29] There is no easy answer.

[28] Wachter, B. 'The overdose: harm in a wired hospital'. Wired magazine (2015). https://www.wired.com/2015/03/how-technology-led-a-hospital-to-give-a-patient-38-times-his-dosage/.

[29] Christie, J. 'The dragons of the unknown; part 5 – accident investigations and treating people fairly'; author's blog (2019).

How to respond

There is no easy answer. It was worth repeating that phrase because it is the first point we have to accept. Building and managing systems that are more reliable in operation and as a source of evidence requires a series of improvements that entail compromises and balancing risks. This requires skilled judgment by experienced practitioners who know that perfection is unattainable, but understands how to reduce the risks of the worst outcomes.

More reliable systems

Although I have argued that reliable programs can combine to form unreliable systems, and that faulty programs can work effectively in a system that is generally reliable, it would be wrong to infer that the quality of the individual programs is anything but vital. That would be nonsense. The better the components, the better are the chances that we will have a good system, and the more confidence we can have in its outputs. Useful systems with faulty components can still be valuable because of the efforts of skilled people to keep the quality as high as possible. It is a constant struggle, but if they stop worrying about the individual components the system will inevitably and quickly deteriorate.

I am reasonably confident that the quality of programs will improve as more developers learn how to produce good code. Greater coding discipline and coding reviews have an important role, as will attempts to produce code more predictably and accurately from program specifications. However, this latter point raises the question of how we can be confident in the quality – not only of program specifications – but also of the requirements and design specifications. It has always been easier to build correctly against the specification than to be sure that the specification was right. Building systems correctly to match our limited imagination of what is required has always been easier than building the right system to fit the real world.

I agree with Tom Gilb, the American systems engineer, consultant, and author, with his argument[30] that system developers usually rush far too quickly into system design before they understand what the system has to do. Requirements specifications describe the high-level design, one of the possible solutions to the requirements, rather than the requirements themselves. The result is a system that does not consistently do what it must do, or prevent what must not happen. I believe that this applies, in particular, to the controls that the system requires.

As I explained at the start of this article, when we conducted audits of live systems, we would start by discussing what the system must do and what it must not do. We never allowed the users we were interviewing to refer to system design diagrams. We wanted them to explain in business terms what the system had to do and to prevent, i.e. the controls the system needed.

Our focus on system controls mirrored the approach taken in safety engineering where safety is considered a control problem. Professor Nancy Leveson is one of the leading experts in that field and has developed a technique for modelling hazards and the system controls that are required, STPA (Systems-Theoretic Process Analysis).[31] In STPA losses are unacceptable outcomes, hazards are events or system states that could lead to losses, and constraints are the behaviours, or controls that the system requires in order to prevent hazards occurring.

STPA modelling is very similar to the modelling technique that we used as auditors and which we found extremely valuable to learn about systems, identify potential weaknesses and gain confidence in them, where appropriate. We asked development teams to follow a similar approach to produce what we called control specifications. We had limited success with this because these requests were always framed as informal advice rather than formal recommendations after an audit. However, control specifications were valuable when they were used. For instance, they provided a starting point against which we could assess the system in an audit. Perversely, however, that never happened because our audits were always risk based, and in practice they were directed at the systems about which we had existing concerns. Unsurprisingly these were ones that

[30] Gilb, T. 'Competitive Engineering', chapter 3 'Functions'. Elsevier (2005). Argument summarised in; Christie, J. 'Tom Gilb and project requirements'. Author's blog (2010). https://clarotesting.wordpress.com/2010/05/19/tom-gilb-and-project-requirements/.

[31] Leveson, NG. Thomas, JP. 'STPA Handbook'. Partnership for Systems Approaches to Safety and Security (2018), https://psas.scripts.mit.edu/home/wp-content/uploads/2013/10/An-STPA-Primer-version-0-4.pdf.

had never explicitly considered the problem of controls.

In devising control specifications, it is important to think beyond the activities and processing flows that usually occupy designers' thoughts. In their paper,[32] Ladkin and others offered an interesting idea in the section 'A 'Third Way' between PACE 1984 and the LC Presumption' on page 8:

> '…there is the question whether an IT system complies (and, if so, to what level of conformance) with any relevant standards to the application that it nominally serves. For instance, consider the bookkeeping system in relation to *Bates v Post Office Ltd (No 6: Horizon Issues*). It is a requirement for commercial bookkeeping to record all transactions. It is also important not to record transactions that do not occur.[33] It would follow that an IT system that performs commercial bookkeeping functions should adhere to these requirements, amongst others. However, it seems that Horizon did not.'

This accords with my experience as an auditor, developer and tester. The simple statement 'it is a requirement for commercial bookkeeping to record all transactions' is so obvious that it appears banal at first sight. Yet this is exactly the sort of obvious requirement that is easily missed by users for the very reason it is obvious – it 'goes without saying'. As I described at the start of this article, in conducting audits this is the level at which we would begin, asking questions about whether processing was complete, accurate, authorized and on time. It was not good enough to say that processing must be complete. The system had to be designed and built so that there was evidence that processing was always completed successfully or that any failure was reported.

One of the Horizon problems that shocked me, with my background in accountancy, was that it breached the rules of double entry bookkeeping (e.g. the Dalmellington Bug[34]). It is completely unacceptable that a single, non-zero, accounting entry can be created without its double, a counter-balancing entry. The system should comply with this principle, with each successive phase, build, or release being carefully designed and tested to ensure the integrity of double entry processing. The Post Office and Fujitsu seemed to lose sight of this until it appears that they lost control, perhaps without even realizing they were no longer in control.

Double entry accounting is just one example of a standard that can be used to build, test and assess systems. Most of these standards or rules will be specific to the domain or the organization. This was an approach we followed when I was a development team leader, building and testing complex insurance finance systems. We were handicapped because our systems produced answers that were not available any other way. The question for us was how could we know they were right?

The lack of readily available external oracles against which we could test meant that any business rules or relationships that must apply to the data and hold true over time, or over a large number of records, gave us something to shape our development and guide our testing. For instance, for a given policy the premium paid on an individual transaction bore no necessary relation to the earned premium. It could even be negative. But over the full length of an insurance contract, the sum of the premiums paid must equal the premium that was earned.[35] You do not need to understand the insurance terminology to appreciate that this is the sort of rule that can be built into the system. If the sum of A does not equal the sum of B then sound an alarm.

These rules were a mixture of business rules and rules that could logically be inferred from the data. Some of

[32] Ladkin, PB. Littlewood, B. Thimbleby, H. Thomas, M. 'The Law Commission presumption concerning the dependability of computer evidence'. Digital Evidence and Electronic Signature Law Review, 17 (2020), 1-14.

[33] That is, the transactions will not be recorded, but the rollbacks ought to be logged.

[34] *Bates v Post Office Ltd (No 6: Horizon Issues) Rev 1* [2019] EWHC 3408 (QB), [445]-[448].

[35] The accounting accrual principle requires that revenues and expenses are assigned to the period with which they are associated, and not necessarily the period in which money changes hands. Insurance premiums have to be earned on this basis. If a policyholder takes out an annual insurance policy and pays the full cost of £365 on 1st December 2020 and the insurer's financial year end is 31st December then only £31 counts as earned premium in the 2020 accounts. The remaining £334 will be treated as earned premium in 2021. The insurer therefore owes the policyholder £334 worth of cover at the year end.

them were arbitrary rules imposed by our design. These might have no business significance, but breaching these rules would mean that we had done something to the data, or unexpected data had done something to our system, that we did not understand. Bugs discovered during testing were valuable. They revealed something that we had failed to understand about the data and exposed false assumptions. We would design the system's processing around these rules. In live running, these checks would identify any deviations. Serious discrepancies meant the run would stop and a developer would get a telephone call in the middle of the night.

Reliable systems should, as a matter of course, produce the logs, reports and audit trails that will serve as credible evidence. We auditors were often asked to specify control reports for a system. We always refused, and insisted that control reports should be specified by analysts interviewing the system owners and users to understand what they needed to control the system. Control reports were not some arbitrary audit imposition, but an integral part of a well-designed and well-managed system. They provide vital evidence about the reliability of the system.

More reliable evidence

More reliable evidence should be seen as a by-product of more reliable systems. Well-managed IT installations will produce good evidence as a matter of course. There were two other interesting and related points made in the suggested third way by Ladkin and others. Courts should consider how likely is error to occur, and whether it is credible that evidence could have been affected by computer error. The example they give of the latter point is a spreadsheet application. It is a good illustration. These are notoriously unreliable because it is so easy to build quick, sloppy spreadsheet applications. I always assume such an application to be faulty unless I can satisfy myself that it has been developed to provide evidence of reliability. When I develop one, I always incorporate controls to highlight errors and inconsistencies. The same principle that applies to a small, simple set of church accounts applies to a corporate accounting system.

Ladkin and others also say that courts can assess the likelihood of error by consulting error logs and security patch records. The absence of either should fatally undermine any computer evidence. I have experience as an information security manager. Any organization without a documented security patch process and records of patches applied should avoid any situation where they might be asked to submit computer evidence to a court. As with a lack of control over superusers, the organization would struggle to establish credibility if they were facing a well-prepared opponent who understood the significance of these failings. Ladkin and others also mention audit records, but only in relation to project management. That is a limited view of what audit can offer. Internal audit should play a large role in demonstrating whether computer systems are reliable.

If the courts are to have confidence in computer evidence, they should expect appropriate, corresponding evidence of system reliability to be available as part of the normal management of systems. The providers of that evidence should face the burden of demonstrating that they are in control, and have taken reasonable steps to make their systems reliable. A common, and entirely correct, argument against the current presumption that computer evidence is reliable is the unrealistic burden it places on anyone who wants to challenge such evidence.[36] I know from experience that probing a complex system for errors requires both expertise and unrestricted access. It is an impossible hurdle for inexperienced outsiders to clear.

Organisations who wish to present credible computer evidence in legal proceedings should therefore be expected to perform rigorous system audits of the type I have described. The resulting reports should be produced in the expectation that they could be disclosed in the proceedings. Such audits would require companies to recruit and train good IT auditors, who would be expected to comply with the exacting professional standards of the Institute of Internal Auditors. Companies who are not able to do this should hire audit consultants on a contract basis to audit critical systems. These are expensive options,

[36] From the legal perspective, the presumption is less of a problem if it is purely evidential, which it is, but that does not make it any easier for the person whom it is directed against from raising sufficient evidence to challenge the presumption. See Chapter 6 in Stephen Mason and Daniel Seng, editors, *Electronic Evidence* (4th edn, Institute of Advanced Legal Studies for the SAS Humanities Digital Library, School of Advanced Study, University of London, 2017).

but there are no cheap or easy answers to the problems of software that is often unreliable and, even when it is reliable, lacks evidence about its quality.

For decades software developers, including testers, have aspired to be more professional and more like true engineers. This aspiration has largely been about improving their image, and has not developed into a realistic programme for improvement. Realism is absolutely fundamental to any improvement. Too many grand initiatives in software development, such as Structured Methods, and ISO standardization, have foundered on their lack of understanding of the true nature of software and the way that it has to be developed.

Everyone involved in developing, testing, managing, and operating computer systems will have to be more honest about the incomprehensible, intractable, complexity inherent in software and the limits of what they can say about complex systems.

Conclusion

In the 1990s, as a development team leader, I forbade one of my programmers from using a clever technique in production programs to generate code dynamically at run time in response to the data. Looking at the source code, one saw only code that would write more code, and there was an incalculable number of possible code variations that might be executed. One could not review the code before it was run, only the set of code that was actually run after contact with the ever-changing live data. My reason for banning the technique was simply that this tipped programs that were confusingly complicated over the edge into being utterly indecipherable, unpredictable, and unmaintainable.

It was possible to insist on more conventional programming to achieve our objectives in those days. In the 2020s, these self-adaptive programs are now commonplace. Programs are coded so that they refine their logical algorithms as they are run. New technology is arriving that will require us to think differently about what accuracy means and how to achieve it. Even traditional systems, developed using conventional techniques, have increased in complexity beyond our ability to describe their behaviour precisely. We have arrived in an age of baffling, incomprehensible complexity. The problem will only get worse unless the IT industry starts to take it seriously and respond effectively. Developers will have to learn greater discipline and professionalism, and, above all, honesty about what they are doing.

The important point – paradox even – is that if we are to say anything credible about software, then we have to admit that we cannot be certain about what it does. Lotfi Zadeh, a US mathematician, computer scientist and engineer introduced the idea of fuzzy logic, a way of developing systems when one is faced with imprecise data and complex problems. As Zadeh said almost 50 years ago:

> ‘In general, complexity and precision bear an inverse relation to one another in the sense that, as the complexity of a problem increases, the possibility of analysing it in precise terms diminishes. Thus ‘fuzzy thinking’ may not be deplorable, after all, if it makes possible the solution of problems which are much too complex for precise analysis.’[37]

This has been distilled into Zadeh’s Law of Incompatibility, usually stated as follows:

> ‘As complexity rises, precise statements lose meaning, and meaningful statements lose precision.’

When we are dealing with complex software systems, we can be meaningful, or we can be precise. We cannot be both; they are incompatible in such a context. It might be hard to admit we can say nothing with certainty, but the truth is that meaningful statements cannot be precise.

This all has serious implications for the law and the conduct of court cases. The true experts might provide informed opinions about probabilities, possibilities, critical assumptions, confidence levels, acceptable margins of error, the availability of oracles against which to assess accuracy, and what may have gone wrong in a stochastic system for particular classes of users, without offering any certainty. In contrast, the poorly informed might appear confident, certain and persuasive as they assert from one perspective the correctness of a system, they do not realise they do not understand. When the court has been primed by a presumption that computers are ‘in

[37] Zadeh, L. ‘Fuzzy languages and their relation to human intelligence’. Proceedings of the International Conference Man and Computer, Bordeaux, France (1972).

order' the danger is that glib superficiality will always win the day without appropriate cross-examination by barristers that are aware of the problem.

It is of particular importance that IT workers and legal practitioners understand the significance of the quote from Gerald Weinberg I cited earlier. 'Quality is value to some person'. System quality is not an absolute attribute that is independent from its users. Quality depends on the needs of the various users. Different user communities might have very different, legitimate expectations of the quality that a particular system should provide. If a system is robust and reliable for one community it does not follow that it is acceptably reliable for others, but this is the implication of the presumption of computer reliability.

Developers have to be able to demonstrate that they have taken all the steps that are reasonable to render the software reliable and comprehensible and that they have reduced the field of their ignorance as far as possible. The aim for system developers should be that they have done enough to feel confident if they are held accountable for how their software performs in the hands of responsible, irresponsible, and even dishonest, users and also for the quality of evidence that their software provides about what has happened.

The IT industry must accept the premise that guided us as IT auditors. Evidence should not be assembled in an ad hoc manner for litigation. Appropriate evidence must be generated as a basic requirement of every new system. Further, the need to produce and retain evidence should be seen as a necessary by-product of a well-managed IT installation.

If system developers are to produce more reliable systems and ones that offer credible evidence on which a court could rely, then they will need to adopt techniques such as Leveson's STPA, which I offer only as one credible example. Developers, responding to commercial pressure, have been too cavalier with the quality of their systems. They will have to take more time to think their way through potential problems and ensure that systems are designed, and built, in a modular way that will facilitate testing and investigation. This will inevitably mean that both the development process and the systems themselves will be less efficient, in the short term at least. In the longer term it will be accepted that a system that fails to meet its most basic control objectives is poor value at any price.

Removing the presumption of computer reliability might lead to better outcomes in legal proceedings, but of itself it will do little to improve the quality and evidence that systems provide. Appropriate reforms may well require regulation to force companies to take responsibility for the consequences of their systems' behaviour, and to protect the responsible from irresponsible competition.

Whatever reforms are needed, they must be based on the premise that computer systems are not inherently reliable, indeed that complex software is inherently unreliable and unpredictable. I have referred repeatedly to the safety critical engineering community, which is notably active in the field of aviation systems. There is no doubting the sceptical realism that they bring to their analysis of the workings of complex systems. The European Organisation for the Safety of Air Navigation (commonly known as Eurocontrol) is an intergovernmental organisation charged with achieving safe and seamless air traffic management across Europe. In 2013 Eurocontrol issued a highly influential white paper, written by leading experts from the safety critical systems community. It outlined a new way of looking at, and working with systems that have grown too complex and intractable for anyone to understand. The white paper set the scene clearly on page 14:[38]

> 'We must accept that systems today are increasingly intractable. This means that the principles of functioning are only partly known, or in an increasing number of cases, completely unknown, that descriptions [of how systems work] are elaborate with many details, and that systems are likely to change before descriptions can be completed, which means that descriptions will always be incomplete.
>
> The consequences are that predictability is limited during both design and operation, and that it is impossible precisely to prescribe or even describe how work should be done. Technological systems can function autonomously as long as their environment is

[38] 'From Safety I to Safety II – A White Paper'. Eurocontrol (2013). https://www.skybrary.aero/bookshelf/books/2437.pdf.

> completely specified and as long as there is no unexpected variability. But these conditions cannot be established for socio-technical systems. Indeed, in order for the technology to work, humans, and organisations, must provide buffer functionality to absorb excessive variability.'

As this white paper made clear, complex systems can only work satisfactorily if humans are able to vary and adapt systems in response to the problems they face. To help and protect the humans, systems must also contain *essential* inefficiencies; redundancy, margins for error, buffers, backups, fallbacks. Accepting this reality introduces a contradiction that is fatal for any presumption that computers are reliable. Systems are reliable only if there is active, expert human involvement. Once that involvement is acknowledged, one must also accept that one is dealing with human frailty, not infallible machines. The systems themselves can no longer be presumed to be reliable, and any evidence coming from them must be justified and open to challenge in the same way as any other evidence from a human source.

This article should not be treated as a lament of despairing resignation. As I have explained, the aviation industry certainly has no naïve illusions of system reliability, but it does have a remarkable record of safety. The Boeing 737-MAX tragedies of 2018 and 2019 have illustrated failings at Boeing that could serve as case studies for the issues Eurocontrol raised, but they are very much exceptions to the general rule of aviation safety. Commercial aviation has become so safe that in 2017 nobody in the world was killed on a scheduled passenger flight. This has been achieved with systems that the industry acknowledges to be intractable. Indeed, such levels of safety are possible only with the awareness of system fallibility permitted by that acknowledgment. Organisations have to understand that systems are fallible if they are to incorporate appropriate, controlled safeguards into their systems.

The presumption that computers operate correctly, and consequently that computer evidence is dependable, was never appropriate. Far from offering reassurance, such a presumption alarms those who have a grasp of the nature of complex systems. The presumption contributed to tragic miscarriages of justice in the Horizon prosecutions. Retaining it will only lead to continuing complacency about the problems of complex software. It is an absurd anachronism based on a failure to understand both computers and software. The presumption must go.

© James Christie

James Christie a self-employed testing consultant with 35 years' IT experience, as a system developer and designer, business analyst, IT auditor, project manager, test manager and information security manager.

https://clarotesting.wordpress.com/about/

jameschristie2020@gmail.com

Appendix

Poll conducted over Twitter by the author on attitudes of IT workers to the presumption of computer dependability.

Only for people with serious IT experience, anywhere in the world (feel free to share).

If someone in a pub told you the common law in England is that computer output produced in court as evidence should be presumed to be correct (this is true), would your first instinct be to…

Say that’s reasonable	2.3%
Explain why it’s wrong	14.4%
Laugh or swear	83.3%

872 votes · Final results

7:15 PM · Aug 17, 2020

https://twitter.com/james_christie/status/1295424052060979200