Computer Bugs in Hospitals: An Unnoticed Killer

M. Thomas and H. Thimbleby*
martyn@cyberliving.uk harold@thimbleby.net
Independent Consultant Swansea University, Wales

February 26, 2018

Abstract

Computers are involved in all aspects of patient care, from booking appointments through to computers in systems that deliver care, such as ventilators, infusion pumps and pacemakers, as well as in computerised decision support systems supporting clinicians. Computers are used in diagnosis and assessment, in MRI scanners, and weighing machines. They control sterlisation, security, and ambulance dispatch. Everybody has mobile phones, email, calculators and medical apps.

It is likely that computer-related preventable error, including cybersecurity exploits, is significant, but more research is needed to quantify its impact. Our own very conservative estimate is that 1,000 deaths per year are caused in the English NHS by unnecessary bugs in computer systems. Regardless of an accurate assessment of numerical impact, though, we should be striving to minimise it, and enabling procurement to choose safer systems.

We show that manufacturers appear to be unaware of bugs. If they — the most technical people involved — are unaware of bugs, then neither clinicians nor incident investigators will be properly aware of computer-related causes of patient harm.

The aim of this paper is to show that computer-related error is frequently over-looked by programmers and manufacturers. In turn, it is over-looked by regulators, by procurement, and by clinicians. It is ubiquitous and remains highly problematic. We show that there are ways in which computer-related harm can be reduced.

We provide 14 specific suggestions for improvement. Improvement will require tighter regulation as well as improved software engineering.

This paper is based on a public lecture, "Computer Bugs in Hospitals," given at Gresham College, London, 6 February 2018.

*Corresponding author

1 Introduction

Computers are the new medical intervention. That's the promise, but a few people are starting to recognise that computer bugs (and poor software engineering more generally) cause and exacerbate patient harms and also harm staff. Computer bugs are avoidable because there is a growing scientific approach to avoiding and managing them, and hence to reduce harms. It is an interesting and worrying story.

This paper falls naturally in three parts, as follows

Section 2 — The scale of error in healthcare and computer bugs (page 3);

Section 3 — Computer bugs lead to patient harm (page 9);

Section 4 — Pathways to improvement (page 23).

2 The scale of error in healthcare and computer bugs

Preventable deaths and harm in hospitals is a serious problem internationally.

A recent study [1] suggests that there are 440,000 preventable adverse events (PAEs) in hospitals in the USA that contribute to the death of patients each year. As a comparison, this makes PAEs a cause of one sixth of all deaths in the USA. The population of the USA is about 325 million and the population of the UK is about 65 million so (by extrapolation and assuming similar healthcare risks in the two countries) the equivalent number of deaths where PAEs are involved in the UK would be 88,000 per year. As a comparison, "only" 1,713 people were reported killed on the road in the UK in 2013 [2]. So, on these estimates, 50 times more people die following PAEs than on the roads each year. There are many more life-changing injuries than deaths and many other injuries that are not life changing that cause suffering and possible loss of earnings — that's a huge amount of trauma to patients, to their families and to the staff who care for them.

A systematic review [3] of specifically medication error estimates 237 million errors per year in England, causing 712 deaths and contributing to 1,081 deaths. The burden of harm from medication error is estimated to consume 181,626 bed-days costing £98.5 million. This review argues that UK and US data are comparable, hence our extrapolation from [1] is justified.

The UK data from the National Reporting and Learning System (NLRS) [4] records fewer serious adverse events than we estimate here. Their official numbers are 10,800 reported events each year that cause serious harm or death to patients. The review [3, p160] suggests that NLRS detects only 7–15% of all incidents, which suggests the UK serous harm figures are closer to 100,000, which is in line with our extrapolation from the US estimates.

There are many reasons why PAEs are under-reported (we discuss some below). The potential scale of the problem and the uncertainties — even disagreements — in the data strongly suggest there should be urgent, independent research to establish the true scale of preventable deaths and serious injuries.

Individual patients may have better or worse experiences than the national estimates. Thankfully most of us have none. One example, reported in depth in [5], analyses a single patient averaging 100 incidents per year over a period of seven years.

Whatever the exact numbers, the figures are surprisingly large.

> ⚠ This paper addresses adverse events that are computer-related, as that is our area of expertise. We believe that research in this area would be very beneficial — it won't just determine the scale of the problem, but it would help solve the problem and improve the health of the nation.

A lot of attention is paid to cancer, diabetes and other conventional killers — typically (and thankfully) patients have a long time to worry about them and whether there are possible cures. In contrast, preventable errors typically kill very quickly and they usually happen to people who are already ill — so they are easy to overlook or to disguise, and sometimes an error may be a mercy to a very sick patient. When errors *are* noticed, it is usual to blame the nurse or doctor, and not recognise that the errors are part of a bigger picture. If we witch-hunt the clinician, this achieves nothing other than taking our eyes off the system problems that induced them to be caught up in the error. If people are sacked for an error, the end result is we can fool ourselves that "errors do not happen here."

It is interesting that death certificates do not mention error as possible cause of death, and very common errors are euphemised as "complications" that do not have to be disclosed. For all these cultural reasons and more, the scale of preventable error hits us as disproportionate, but there are other ways to conceptualise the impact ...

According to the National Audit Office [6] there were 10,600 negligence claims registered with NHS

Resolution under its Clinical Negligence Scheme for Trusts in 2016–2017 (ten years earlier the number was half that, at 5,300). The cost of these claims totalled £1.6 billion. The NAO expect this to double by 2020–21. NHS Resolution's Clinical Negligence Scheme for Trusts has made a provision of £60 billion to pay for future claims of clinical negligence. These are resources that could have been spent on better healthcare — and more dependable computing for healthcare.

It is likely that only a small percentage of injured patients or their families claim, for reasons that include:

- because they don't notice the error as such, or
- because they don't know it was preventable, or
- because they may become incapable to report it (perhaps dead), or
- because they don't want to make a claim, or
- because there are barriers to them doing so, including the difficulty of obtaining evidence and clinicians' [7], hospitals' and manufacturers' understandable reluctance to admit to being caught up in error.
- Computer bugs, ironically, could be another reason: the computer system used to report incidents may be faulty or too hard to use. Indeed, we both tried using the NHS incident reporting web site on 28 January 2018; we got errors and were unable to proceed.
- There is no standard practice for recording harm when computer systems do not work. It has been reported that WannaCry (see section 2.3) did not cause harm [8], but computer systems were down and it would be hard to attribute harm when a patient was not seen by anyone. Similarly the nation-wide NHS IT failure in Wales (section 4.2) caused no recorded harm. In any case, when the computer systems were repaired, it is likely that staff were busy catching up on their backlogs rather than trying to report complex problems.

All of which means that the future costs might be much, much higher. We rightly care deeply about road safety, so it is astonishing we are doing little about a problem that may be at least 50 times worse and much more costly to the nation and to individuals.

2.1 Computers and healthcare

Healthcare is increasingly dependent on computer-based systems. Computers are used to schedule and book appointments, to order and manage stocks of drugs and thousands of items of equipment, to manage medical records, to optimise the use of beds, operating theatres, scanners and other facilities, to control electronic medical devices, to monitor patients' vital signs, to diagnose diseases and conditions … and much more. In addition, so-called embedded computers are inside almost every medical device, such as infusion pumps, incubators, MRI scanners, X-ray machines, dialysis machines, pacemakers, … and even *inside* pills [9]. And, of course, almost everyone has mobile phones and apps, using computers to run calculators, medical apps and communications systems (such as WhatsApp, as well as conventional phone communications).

Within hospitals, there are thousands of computer systems, mostly networked together in unfathomable ways, and even with special systems ("middleware" — often configured in *ad hoc* ways) to convert between one and another system. The variation in standards across the various systems results in the recognised problems of multiple logins for clinicians (slowing them down, and often leading to insecure workarounds) as well as lack of interoperability for the patient data. Clinicians find it easier to photograph screens than to use the systems to contact colleagues — thus undermining basic information governance (they end up with patient data on their personal phones).

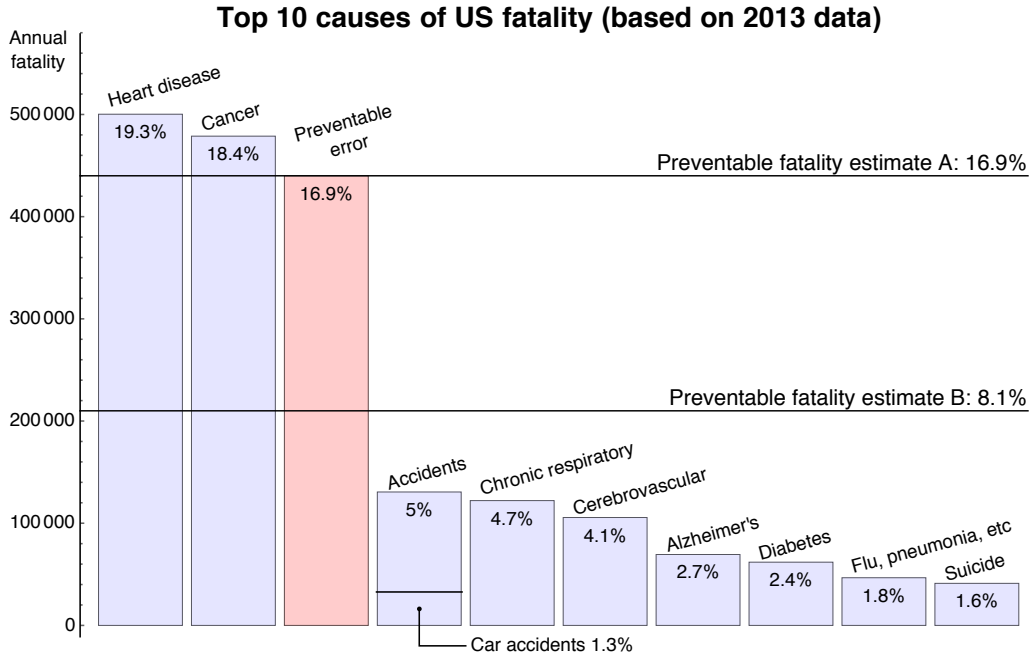

Figure 1: **Bar chart of leading causes of death, including estimated fatality from preventable harms in hospitals. The vertical axis is US fatality. Percentages of total fatality are shown in each bar: percentages are likely to be comparable across developed countries. (Other, less common, causes of death are not shown.) Assumptions for estimates A and B, and further details are provided in the appendix.**

The leading causes of Preventable Adverse Events are errors such as [10]:

- delayed treatment or the wrong treatment,
- no treatment,
- errors of communication perhaps leading to misdiagnosis or operation on the wrong site,
- errors of context — perhaps leading to discharge in circumstances where the patient will be unable to follow their care plan, or
- diagnostic errors.

Errors of omission and intentional errors [11] are hard to notice and may be under-reported. We suspect, too, that computer related errors are under-reported because clinicians are not trained to recognise them.

Because it is easy to envisage examples where a faulty computer system could cause or contribute to each of these PAEs, and a better designed computer could predict, detect, prevent, or mitigate a PAE, it seems reasonable to assume that computer system defects and deficiencies actually contribute to a proportion of the harm. Computer systems support — or undermine — every stage of every care pathway, from the initial GP appointment bookings through to the controlled injection of a drug.

2.2 Published data on computer-related healthcare incidents

A paper in the *Journal of the American Medical Informatics Association* [12] describes several published studies of the risks to patient safety from the use of computers.

- In February 2010, the US Food and Drug Administration (FDA) reported receiving information on 260 incidents with potential for patient harm including 44 injuries and six deaths [13].
- In 2006 almost 25% of 176,409 medication errors reported to the United States Pharmacopeia voluntary incident reporting database were computer-related [14].
- In contrast, a search of 42,616 incidents from 2003 to 2005, submitted to a voluntary incident reporting database across one Australian state, yielded 123 computer related incidents. After removing duplicate and unrelated incidents, 99 incidents describing 117 problems remained. Only 0.2% of all incidents reported were computer-related.

The authors of the paper say "it is important to note that incident reports do not yield true frequencies of errors or adverse events because they do not capture numerators or denominators, and are subject to bias from a number of sources." In particular, few clinicians are technically trained and able to identify computing problems, so we expect significant under-reporting of computer-related errors. In particular, what may seem to be "use error" is often caused by underlying computer error.[1]

We are faced with a spread of computer-related incidents from 0.2% to 25%, the likelihood of under-reporting, the likelihood of bias, the difficulty of recognising harm that may only develop slowly and over an extended period, and the difficulty of attributing causality. Nevertheless, it is clear that a significant number of adverse healthcare incidents do occur that involve the use of computer systems, that many of them have the potential to cause harm, and that some of them have contributed to or caused avoidable injuries and deaths.

That there are errors and harm in healthcare on the one hand, and computer bugs on the other, does not *prove* improving computer systems will improve healthcare outcomes — though it raises the obvious, and urgent, research questions to find out.

While there are suggestive case studies, there has been virtually no systematic research to study any causal relation. We note that the research by Han *et al* [15] showed a more than doubling of patient mortality in a pædiatric ward after introducing a computer system.[2] While it could be argued that this only shows that a particular system used in a particular medical speciality was poorly configured (or was in some way otherwise inadequate) and thus no general conclusions should be drawn, on the other hand, the system was provided by a major supplier, it was not a new and untested system, and the hospital itself was not naïve. Yet serious problems occurred because of the computers; indeed, the paper argued that staff time lost to using the computer (which was therefore a loss to direct patient contact) was not compensated by improved efficiency.

A recent 2017 study compared two EHRs (electronic health record systems) [16] and found one caused over three times as many errors as the other, and twice as many clinicians using the better system made no errors at all (for the specific tasks studied). Put another way, the worse EHR has design bugs that cause preventable error; in fact 70% of errors are caused by its worse design or would have been prevented by using the other design.

A very different study of hospital ambulatory care [17] shows doctors spend 37% of patient time using computer systems, and for every hour they have direct contact time with patients, nearly 2 additional hours is spent on computers and desk work. Unfortunately there is very little unbiased, rigorous research available: hospital computer systems cost billions and require major upheaval to implement, there are financial incentives to believe they are more successful than perhaps they are, and for complex reasons "gold standard" methodologies such as randomised trials are effectively impossible [18].

[1] We prefer to say *use error* meaning an error occurring during computer use, rather than the popular term *user error* which prejudges the issue making it seem like an error made by the user. This paper is about computer bugs: some user errors are caused by computers not by users.

[2] Mortality rate increased from 2.80% (39/1394) before to 6.57% (36/548) after implementation ($p < 0.001$); the ratio is 2.35. See Han *op cit* for details.

In short, to be effective, computers must provide added value greater than the lost time and resources needed to use them *at the point of care*.

If we conservatively assume that only 1% of all preventable adverse events involve computer system defects and inefficiencies (and that computer-related PAEs are as likely to cause serious injuries or deaths as other PAEs), that would mean that computer system defects contribute to 880 deaths annually in England, to far more injuries, and to £16 million of the annual financial liability to the NHS and £600 million of the financial provisions for future negligence.

Unlike almost any conventional intervention (like a new cancer drug), improving the safety of computer systems would *save* money, both direct costs and litigation costs and insurance premiums.

2.3 How computers contribute to death and injury

In healthcare — despite the surprising statistic that the NHS is the world's largest purchaser of fax machines — computers are found everywhere. Computers are in pacemakers and in infusion pumps, in MRI scanners and in operating theatres, in the systems that hold patient records, and in the many other computer systems that support medical staff and that monitor, diagnose and treat patients. A typical NHS trust has more than 150 different types of computer systems for administration, communication and professional support, plus many more that are embedded in medical equipment. Most of these systems have safety implications, and all are vulnerable in some way to cyberattack.

The risk of serious cyberattacks has been assessed as a Tier One threat on the National Risk Register. The scale of the threat from cyberattacks depends on the strength of the attacker's motive (probably in proportion to the damage they intend to cause) and the ease with which the attack can be carried out. The head of the UK National CyberSecurity Centre said in January 2018 [19] that a serious cyberattack on the UK was a matter of "when, not if." That warning applies equally to healthcare too.

The groups that carry out cyberattacks are (in rough order of capability) teenage vandals ("script kiddies"), single issue activists ("hacktivists"), minor criminals, terrorists, organised crime groups, and even nation states. Kits are readily available that enable unskilled hackers to build new attack systems.

In the decade since one of us was on the Board of the Serious Organised Crime Agency, SOCA (now the National Crime Agency, NCA), we have seen the cyberattack tools and methods that were formerly only used by nation states migrate through organised crime down to ordinary criminals, hacktivists and script kiddies. One example is the EternalBlue exploit that was developed by the US National Security Agency's elite Tactical Access Operations team, then stolen by a hacker group, The Shadow Brokers, and published on the internet along with dozens of other exploits and tools — over a gigabyte in total. Once published online by the Shadow Brokers as a toolkit, what was once an obscure technical break for the NSA became within the competence of any basic hacker to use.

That exploit was used in the WannaCry ransomware attack that affected 200,000 computers running Microsoft Windows, including some in the NHS [20, 21]. Fortunately, the objective of the criminals who wrote WannaCry seems to have been criminal extortion rather than terrorism, as the software announced itself with a screen that informed the users that all their data had been encrypted and that the way to recover it was to pay a ransom in Bitcoins. Even though the NHS was not a specific target of the WannaCry attack, 37 hospital trusts (including 27 acute trusts) were infected and locked out of devices, almost 20,000 hospital patient appointments were cancelled, 44 hospital trusts were not infected but experienced disruption, 21 trusts and 71 GP practices had systems trying to contact the WannaCry command server (but were not locked out of devices), 595 GP practices were infected and locked out of devices, plus there was an unknown amount of further NHS disruption [22]. The

disruption would have been worse had WannaCry not been stopped by a cybersecurity researcher activating a "kill-switch," albeit largely by chance.

Fortunately, WannaCry's disruption to healthcare, though significant, was limited. But the WannaCry software could easily have been designed to make changes to critical data instead of encrypting it, because if files can be read and encrypted then, clearly, they can be changed arbitrarily. Imagine what the effect would have been if the WannaCry software had instead been designed to change critical fields in medical data and if, rather than announcing itself with a ransom screen, it had remained hidden until several backup cycles had passed. How quickly could the NHS recover if blood groups, allergies and other life-critical fields in electronic medical records could no longer be trusted? Recovery from backups can also be thwarted with a bit more thought.

- That WannaCry was able to encrypt healthcare data raises serious questions about certification: the vulnerable systems were unreliable. How can a system be certified when it can be adulterated? It is unverifiable. Hospitals should rely on effective monitoring and backup processes and manufacturers should make systems that are capable of being managed reliably.

A lot of hospital equipment is attached or could be attached to the internet, including many systems that have safety-critical roles in treating, diagnosing or monitoring patients. Attacks can change the drug delivery rates on infusion pumps remotely, or even reprogram someone's heart pacemaker from a few yards away and using bluetooth.

The scale and complexities of the cybersecurity healthcare problem is illustrated by the St Jude case. A security company, MedSec, revealed cyber-vulnerabilities they had found in a St Jude pacemaker. This resulted in the US FDA requiring 465,000 pacemakers to be reprogrammed to fix the bug [23] — these are pacemakers inside patients, who would need to visit a competent hospital for the reprogramming. In a twist, MedSec collaborated with Muddy Waters Capital who sold St Jude shares short to profit from the anticipated fall in the share price. This, MedSec argues, was to fund their business of finding security flaws.

Evidently, critical healthcare systems are being developed with insufficient thought for managing cyberattack threats. Often, they use software components that were designed for other purposes. They may contain hidden services with default passwords that the user cannot change or does not know about. If there are support arrangements at the beginning, they are likely to end before the device is replaced, because (as Boston Scientific explained to the FDA at a workshop [24]) manufacturers continue to use off-the-shelf software with a 3 to 7 year lifecycle in devices with a longer than 15 year expected life [25]. When support ends, these devices remain for a decade or more with all their vulnerabilities. They are an open door that may expose connected systems and networks to uncontrollable risks. WannaCry is an example of this problem: it infected MRI scanners that were running obsolete software that was no longer being maintained.

Medical equipment is not designed to the same standards as some other safety critical systems (for example, flight-critical aircraft systems and ground-based air traffic control). Professor Nancy Leveson has described in detail the defects in the system design and the software of the computer controlled Therac 25 radiation therapy machine that massively overdosed and killed patients between June 1985 and January 1987 [26]. That was well known (at least to professional computer scientists) over 20 years ago, but there is still little evidence of improvement in the quality of software and system design or in the standards that regulators require before medical equipment can be used with patients.

The most significant regulator, the US Food and Drug Administration (FDA) regulates medical devices, but it does not verify (or even test) the performance of individual devices because, as it says,

> "The FDA does not conduct premarket testing for medical products. Testing is the responsibility of the medical product manufacturer" [25]

The FDA reviews the documentation that the manufacturer provides to justify their device being approved. For decades it has been sufficient to provide a demonstration to the FDA that a new device is "substantially equivalent" to a device that has already been approved — the so-called 510(k) process. In 2011 the US National Academies of Science reviewed the 35-year old 510k Process and found it to be ineffective in assuring safety or clinical effectiveness. With regard to software, they concluded:

> "Manufacturers are increasingly using software in devices, software as devices, and software as a tool in producing devices. That trend is expected to continue. The committee found that current guidance on software validation is insufficient for preventing serious software-based device failures" [10]

The FDA is now recognising that the regulatory process for computers will have to change, and they are considering radical changes [27]. In the UK and indeed across the EU, the situation is similar, and CE marking of algorithms and programmed medical devices is currently self-assessed [28], which means that the team that overlooked bugs in the first place may be the same team that self-assess to certify the software is safe. This is an unsafe process. Or it is assessed through documentation rather than analysing source code and specifications. This is a hopelessly obsolete approach, and takes no advantage of software tools to help automatically verify compliance.

We have explained in previous Gresham lectures why it is impossible to justify high confidence in any software-based system if your main evidence comes from testing, and we have explained how it is possible to write software so that you have strong evidence that it is correct, safe and secure [29]. Few manufacturers of medical devices develop software in this way and no regulators demand that they do. Until this changes, the risks to patients from cyber vulnerabilities and other defects in medical devices will remain unnecessarily high.

While cyber-vulnerabilities have captured huge media attention, general software defects are a bigger problem, not least because they do not attract attention. They cause continual, everyday problems that are going unnoticed.

3 Computer bugs lead to patient harm

Many healthcare computing projects have failed or have had disappointing outcomes: healthcare computing is widely recognised as a vexing problem [30]. The US National Academies calls it a "healthcare IT chasm" [31], and Ian Foster says, "healthcare is arguably no longer a medical problem, but a computing problem" (cited in [32]).

We would express the issue differently: healthcare computing is the first time that the rules of healthcare have had to be written down in any form as precisely as computers require, and this new rigour follows thousands of years of informal heuristics and flexibility being culturally embedded in healthcare. What computers are now doing is clearly showing the incompleteness and unsoundness of the combination of medical practice — and regulators and politicians imposing complex managerial demands and performance targets on the systems. There are disparate, unstandardised methods across different specialities. We have an incomplete understanding of disease and patient care, unformalised rules of care, clinicians work in different ways in different specialities (and they may invent new procedures to meet specific patient needs), and programmers poor have a poor grasp of these complex details and their interactions. The systems they build have to network with (interoperate with) a huge variety of disparate systems. Previously it did not matter much if radiology and oncology had different procedures and even different terminology, but when they both use networked computers, they have to be the same and understood the same ways — or else confusion ensues. You cannot mislead a computer and expect it to work reliably!

On top of all that, programmers may introduce bugs.

Fig 2a

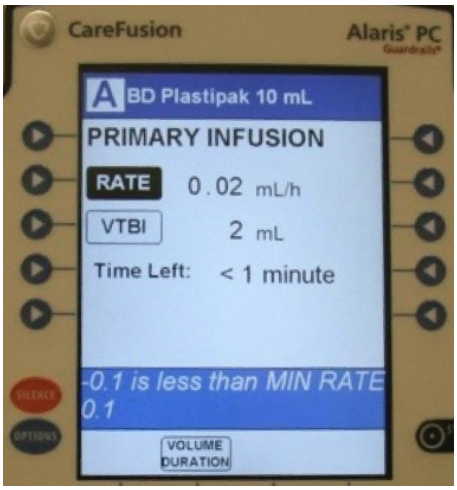

Fig 2b

Figure 2: **Very simple bugs to see, with preventable consequences. (a) The computer-printed label has unnecessarily truncated the doctor's name and where he works, though it had no problem with the patient's long surname. The doctor's surname is *not* Jon, and he does not work in a Medical Ce! If a name *has* to be truncated, it should be displayed as "Dr Richard Jon..." making it clear something is missing, or the first name could have been abbreviated, perhaps down to an initial (which is conventional practice). One day, items labelled incorrectly will be misplaced or used with the wrong patients. (b) The infusion pump is warning that the set rate of 0.02 mL/hr is -0.1 and less than 0.1. The pump required rebooting before continuing.**

3.1 What is a bug?

We have chosen to use the colloquial term "bug" rather than "defect" or "error" to emphasise the medical analogy with infections. In hospitals, biological bugs are recognised as a major threat — the damage they can cause is prevented where possible through rigorous hygiene and actions to enhance the immune system, and there are protocols, such as blood tests, in place to detect and destroy them with powerful remedies where necessary.

We argue that the same is essential for defects ("bugs") in digital systems: strict professional rigour to prevent them being introduced, barriers (firewalls) and high standards of hygiene to stop infections from outside (cyberattacks) and powerful software engineering tools and methods to detect and eliminate those that penetrate and survive to contaminate our medial devices and digital systems.

Some people differentiate different sorts of bug, a concern beyond the immediate scope of this paper. Too often the cliché "it's just a bug" makes us think they are trivial and nobody's fault. In fact, bugs are caused by humans. Behind them there are many types of human error [11], e.g., slips and intentional errors as well as rush-to-market, overwork, under-resourcing, incompetence, negligence, even hubris and crime ... all of which lead to bugs.

We use the term "bug" in this paper precisely. Programs have requirements, and the programs should correctly implement the requirements (which is something that can be — and should be — formally proved). In figure 2 we show a program that has failed to correctly print a doctor's name. A failure to implement the requirements is bug. If there were no requirements, then no correct program can be written, and hence no requirements itself is a bug.

Often requirements have implementation bias — they are too specific. The programmer may do what is required, but what is required is wrong. This is a requirements bug.

Sometimes requirements are vague, inconsistent, unsound or incomplete, or just out of date. Attempt-

ing to program requirements that do not work inevitably results in programs with bugs.

Sometimes bugs are unavoidable; healthcare is complex and the first things implemented may not be adequate however well they were designed. The main solution is to use iterative design [33]: systems must be tested and improved. Of course, programming lends itself to this, as upgrading software to fix bugs and make other improvements is routine. Improving systems also requires manufacturers to monitor how the systems are used.

Bugs cause problems. We emphasise that almost always they are preventable and can be avoided by competent programmers using modern software engineering processes [34–36]. Further ideas are discussed in section 4.1 below.

3.2 Programming in a complex system

A successful company like Amazon, Apple or Facebook can create themselves out of nothing, and computerise everything they do. And they do it very successfully — creating the impression that computers make businesses very efficient (though note that for every Amazon there are thousands of companies that failed to get their computing systems working well enough).

In contrast, healthcare computing cannot start from scratch and "do it properly"; it has to make a large messy organisation a bit better. Healthcare has to cope with politicians and regulators imposing additional rules (such as performance targets, information governance rules, auditing requirements, security and passwords, and more) on computer systems, and these are unlikely to be consistent (in the sense a computer requires) with all the details of what is *actually* going on. Unfortunately, since computer manufacturers would like us all to believe that just buying a new computer will solve all our problems, healthcare tends to fill up with new "solutions" that only create more complexity.

Rather than despair at the scale of the problem, a better approach is to make healthcare computing safer and more efficient to use. Take the car as an analogy. Although there are many disparate computers inside cars, the driver does not need to worry about them or how they work. The brake pedal slows the car down. That it does so using ABS and controlling the car's throttle, even supplemented by proximity detectors to help avoid collisions automatically, is immaterial to the driver. We have standardised what brake pedals do and the underlying complexity is hidden. Healthcare computing is a mess because we have failed to agree, let alone standardise, what we are trying to do, and therefore "how things work" (or fail to work) is painfully visible at every angle. It is like having lots of different brake pedals because everybody can think of a nice feature to add, but nobody is willing to simplify and improve what we already have and make it work seamlessly.

3.3 Healthcare examples

Defects in user interfaces are unwelcome news to both developers and vendors. The EU regulatory structure ensures that manufacturers are not responsible for use error if their system has CE marks, and CE marks are a weak standard; similarly, in the US since Riegel *v* Medtronic 2008, the medical device companies have immunity from liability for almost every use of a product if the FDA has approved it. Ironically, vendors claim that regulation (such as it is) inhibits innovation. Manufacturers who make air bags and cars do not claim regulation stiffles innovation; they prefer to make safe cars.

We discussed the impact of preventable errors above. In our laboratory, we have been interested in *unnoticed* errors, which are errors that occur during use of a computer, but which nobody notices at the time. Thus they can lead to preventable errors: with unnoticed errors, some more dramatic consequence, like patient harm, has to occur before anyone realises an error has occurred. Hence unnoticed errors are not corrected until, sometimes, unfortunately, it is too late ...

Entering numbers is a very common task in hospitals, for instance to specify drug doses. One way to enter numbers is to use a numeric keyboard (i.e., using digits 0123456789 and •, the decimal point); another common way is to have two buttons like △ and ▽ to adjust the number "up" or "down." (We discuss some variations on this basic scheme below.)

In our work [37, 38] we have shown that designs with up/down keys lead to about half as many unnoticed errors as those with numeric keypad input. This research is significant:

- Where numeric keypads are used, unnecessary patient harms may be caused not by nurses or other users but by the designs. To put it strongly, about half of the harms caused following use of numeric keys are caused by poor design.
- Unfortunately, device logs are inadequate to distinguish design error from use error (partly because of bugs, and partly because they typically record only what the device did, not what the user actually did or intended).
- Hospital and other procurement should take careful note of research results, and would thereby be able to reduce harm, improve patient outcomes more generally, and also reduce consequential litigation costs. *Research into safer use of computers can save the NHS (or any healthcare provider) money.*
- It is beyond the scope of this paper, but our research also explains *why* we get these improvements [37, 38]. The explanations could be used to drive improvements for many types of computer system.
- Clearly, more research could deliver further useful insights — funding is needed both for research and for translating the research into improved care and performance for healthcare as soon as possible.

In addition to finding differences in how different designs are to use, we have also shown that most numeric user interfaces have numerous bugs in their software [39]. Many publications, such as [35,36], show we can formalise relevant user interface properties, check them against regulatory and other requirements, and hence avoid such design defects almost automatically through using formal method development tools. In other words, bugs are avoidable, and are generally the symptom of poor design practices. Interestingly, as up/down keys are simpler they are often more reliably programmed.

Sometimes numbers are entered using four arrow keys, where up/down arrow keys increase and decrease digits, and additional left/right arrow keys move to which digit to adjust. For instance, to change 10 into 20, the user would move left, then press up to increase the 1 to get 20. This style of user interface is very reliable from a user point of view, because it forces the user to closely watch the display (as the basic up/down design does) and hence be aware whether the number actually entered is the one they intended. Unfortunately, up/down/left/right is rarely correctly programmed.

On the leading B. Braun Infusomat infusion pump,arrow keys move a cursor and can adjust a digit to set drug doses. In some cases, a digit *not* under the cursor is changed, which can lead to number entry errors with no warning (we found this with software version 686E). This is clearly a bug.

On a Zimed AD syringe driver, the arrow keys "wraparound" and thus a single right-move from the highlighted 0.**0** cursor position is not blocked but goes "round the back" to the leftmost **0**00.0 position (i.e., the cursor is placed over the hundreds digit). Consider the user trying to enter 0.01 using the natural sequence of keystrokes to move the cursor right from 0.**0** and to increment the digit (expecting to get 0.01): the sequence will in fact get **1**00.0, which is 10,000 times higher than expected — and with no warning from the Zimed.

Since wraparound like this is pointless, the risk of this bug can easily be eliminated once identified, and such improvement in safety can be done systematically [40].

On the Baxter Colleague 3 infusion pump, drug doses are entered using a numerical keypad. So if the nurse enters 10.5, then a dose of 10.5 mL per hour may be delivered to the patient. Unfortunately if a nurse enters 100.5, the Colleague ignores decimal points in numbers over 100, so a number entered as 100.5 is treated as 1005, even though the decimal point keep "clicks" when it is pressed, providing misleading feedback to the user.

Similar avoidable bugs have been found in almost every computer-based healthcare device we have examined. Don Norman's classic work on programming away use error was published in 1983 [41], and it is astonishing his work and the research that followed is so widely ignored; unnecessary bugs persist.

Of course, when bugs are demonstrated step-by-step, as above, the bugs often seem trivial — surely someone would notice a number 10,000 times out from what they expected? No. In a typical pressurised clinical situation, everyone has a much more important job to do than debugging software; these simple problems can easily escalate to serious harm. Contrast the hectic clinical world with the different world programmers live in: they have years to avoid bugs, they have many tools that can help them avoid bugs test out and improve systems, and they do not work under life-threatening distractions. Programmers typically do not think users make mistakes (errors, slips ...); in fact, concentrating on their programming is hard enough without thinking about users. Our culture — professional clinical bodies and the media in particular — assumes clinicians are perfect; the resulting mix unfortunately does not often think about error and ways to detect, block or mitigate it.

Our examples above were chosen to be accurate and straightforward to explain; the examples do not rely on detailed understanding of any clinical context or pathways: they are "just" numbers going wrong [39]. As well as infusion pumps like those just discussed, hospitals have much more complex computer systems too: such as order entry systems, MRI scanners and more. These have bugs too, but are somewhat harder to explain. Schiff [42] studied ten hospital order entry systems, and found an astonishingly wide range of serious — and unnecessary — problems; their report is unusual for including screenshots of problems. The Schiff report also includes recommendations, and should be required reading.

3.4 Key bounce

When a button is pressed on an electronic device, small conductors move to make an electrical connection, which is then recorded as a key press. Unfortunately, the conductors usually bounce, perhaps 100 times in a millisecond before they settle down. Key bounce is a standard problem, and it must be solved for buttons to be reliable. A simple solution is to use some electronics, but it is cheaper to connect the button directly to the computer, and sort out the key bounce in software. If the programmer forgets about key bounce or programs it incorrectly, the program will have a bug.

Alaris was issued warning letters by the US FDA in August 1998 and October 1999 outlining key bounce problems with the Alaris SE infusion pump [43]. Then in 2006, the FDA issued a Class 1 Device Recall (Class 1 means there is a recognised risk of death) for the pump, because it had uncorrected key bounce. Their recall affected about 150,000 devices in the USA, and involved US Marshals seizing equipment from the manufacturer worth $1.8 million — the FDA did not seize from hospitals, as that would have put patient treatment at risk. One example reported was a patient who received an over-infusion of oxytocin [44]: the infusion pump was intended to be set for 36 mL/hr but was set to a rate of 366 mL/hr, ten times higher — in other words, the digit 6 bounced, and was recorded by the pump as two presses, effectively making 66. If, after a key bounce like this, there is patient harm and an investigation, the infusion pump's log will appear to show that the nurse entered (in this case) 366 mL/hr, making it look like the nurse entered a ten-fold over-dose. In fact, the device malfunctioned.

We do not know about the seized SE infusion pump, but many devices provide a key click to provide feedback to the user when a button is pressed. When a button is pressed, the click confirms to the user that they have pressed the button once, and that they pressed it hard enough to work properly. However, to make the sound of a click audible, the sound must last about a millisecond, so the sound generation itself can misleadingly conceal underlying key bounce bugs from the user. The Baxter Colleague (section 3.3) is an example of a device that key clicks even when a button is ignored.

The bug is simple and in principle simple to avoid and simple to correct, yet Alaris failed to take appropriate action for 8 years. In the recall letter Alaris finally issued, Alaris informed customers that they would provide a warning label for the pumps and a permanent correction for the key bounce problem once they work one out. A warning label converts a bug into a user problem: the user is made responsible for using buggy equipment correctly!

3.5 The case of QRISK

We compared different ways of entering numbers such as drug doses or patient data above (section 3.3); it is an important topic as numbers are central to healthcare — for drug doses, blood pressure, heart rate, and more. We showed [37,38] that up/down keys are more reliable for entering numbers, and interestingly, they are usually implemented without bugs because they are so simple. Up/down keys just increase or decrease a number, and programming them correctly is easy. In contrast, numeric keys require more thought to implement correctly, and they are very often implemented incorrectly [39]. There is, then, a double whammy: numeric keypads are harder to program and have more bugs *and* errors using numeric keypads are harder to spot when they are used.

QRISK is a calculator used to work out a patient's risk of heart attack or stroke. Various versions of it can be used on the web from `qrisk.org`; you fill in a form with your age, post code, ethnicity, smoking habits and so on, and it calculates a risk. The risk is then used to prescribe statins or to help give you lifestyle advice.

In 2016, the *British Medical Journal* reported that a bug in QRISK may have led to incorrect prescribing of statins to thousands of patients [45]. According to journalists, up to 270,000 patients were affected [46].

We tried QRISK in 2016, and found that its user interface had bugs. It ignored use error, such as typing several decimal points in a number. Thus, QRISK ignored the error in a user keying in a number like 200••5. (We have used a font so that the double decimal point is very easy to see. Usually it is tiny!) Here the user has clearly made an error, and therefore any result is in doubt as would be is based on invalid data. QRISK should have alerted the user to the error so it could be corrected — we know from our experiments that users are unlikely to spot this error themselves. This is a simple bug in QRISK; it failed to validate user input — though calling it a "simple" bug does not mean that its consequences are simple, rather it means the bug should have been simple and indeed standard practice to *avoid*.

Fortunately the 2018 version of QRISK has been updated (perhaps because developers were aware of our 2017 paper's criticisms [39]), and it now warns the user if there is a decimal point error and other similar data errors. The current version does not produce any result that might mislead the user due to data errors (so far as we can see).

However, QRISK still has user interface bugs: using the current 2018 version (QRISK®2-2017), if you press RESET, your estimated risk of heart attack or stroke in the next ten years is reported as 12.3%. Yet this is a risk calculated on no data! In particular, if you had entered correct data but accidentally pressed RESET, QRISK will not warn you the prediction is wrong and based on a cleared form.

Interestingly, when the form is RESET, the data is not replaced with nothing (which QRISK might have spotted) but it is replaced with valid data — it is filled in for a 64 year old, white, non-diabetic, non-smoker, ... As we pointed out in our paper [39], user interfaces can (and in this case, *should*) distinguish between real data (like the ethnicity being white) and no data (like the user not have specified any ethnicity). QRISK does not do this, and therefore it cannot tell the user if they forget or omit to specify an important parameter; and since RESET always sets some data, QRISK cannot tell the difference between the user missed setting data, which would be an error, rather than choosing the default which is not an error. Perhaps worse, if QRISK is used by a GP to assess several successive patients, perhaps one patient's data gets accidentally used for the next patient.

In fact, the original problem reported in 2016 was not caused by a user interface bug: QRISK takes 15 patient parameters, and when these were filled in automatically from patient data, the data was mixed up [47] (apparently by a third party), so QRISK's calculations were based on incorrect data.

QRISK publishes its algorithm in open source form. The algorithm's parameter names are

$$\texttt{age},\ \texttt{b}_{\texttt{AF}},\ \texttt{bra},\ \texttt{b}_{\texttt{renal}},\ \texttt{b}_{\texttt{treatedhyp}},\ \texttt{b}_{\texttt{type1}},\ \texttt{b}_{\texttt{type2}},\ \texttt{bmi},\ \texttt{ethrisk},\ \texttt{fh}_{\texttt{cvd}},\ \texttt{rati},\ \texttt{sbp},\ \texttt{smoke}_{\texttt{cat}},\ \texttt{surv},\ \texttt{town}$$

These are not mnemonic and are not documented in any way, which professional programmers would consider a bug in itself. The algorithm does no data validation whatsoever, so an unnoticed mix up is perhaps unsurprising. Fortunately, when the bug came to light it was possible to compare risk factors based on correct and incorrect data, and hence warn all GPs who had patients with a significantly changed risk score.

Finally, it is interesting that QRISK has been validated, published in a peer-reviewed paper in the clinical literature [48], but the validation did not look at how QRISK was used; it was a purely numerical validation, assuming all the data to QRISK was correct. Thus the clinical paper, in overlooking to validate clinical use of the algorithm, overlooked critical bugs.

As we say elsewhere here, we are using QRISK as an example you can check. There is nothing unusual about QRISK; its problems are common across many hospital computing systems. Since the QRISK system we reviewed above is accessed on the web, it is likely that it will quickly be updated and perhaps these bugs will be fixed. That is a positive side of using computers: if the developers want to improve things, they can quite easily.

3.6 A "Blundering nurse"?

Eighty year old Arsula Samson had a heart attack after she was given an overdose of potassium chloride. The *Daily Mail* paper reported the incident under the headline,

> "Mother-of-four dies after blundering nurse administers TEN times drug overdose." [49]

The nurse pressed the 100 mL per hour button instead of pressing the 10 mL per hour button, so setting the infusion pump to a rate ten times higher than intended. As we showed above, these sorts of error are easy to make and are hard to spot, especially if the infusion pump does nothing to help, such as blocking to wait for the user to confirm such a high dose for a dangerous drug.

The report goes on to say:

> No error was found with the infusion pump and investigators ruled the death was due to "individual, human error."

That "no error" was found with something does not mean it is bug free. It certainly does not mean it has no design problems! How are they sure? (We discussed above the inadequacy of testing; see

section 2.3 and reference [29].) It could mean that the investigators are as blind to bugs as everyone else is.

But a very interesting comment was,

> A Trust action plan after the death saw new infusion pumps and software that reduce the risk of error brought into all wards, medical staff retrained and warned over the dangers of potassium chloride and advice on the importance of a second nurse witnessing medication being given.

Isn't this strongly suggesting that a contributory cause of the death was the poor design of the original infusion pump, combined with lack of adequate training and suboptimal practices? Moreover, the cause was recognised by the hospital, which begs questions of their responsibilities prior to the incident.

3.7 Dr Bawa-Garba

On his admission to hospital, the 6 year old patient Jack Adcock was at risk of death (quantified by the court in the range of 4–20.8%) and he later died of sepsis. Dr Bawa-Garba, his paediatrician, was convicted of manslaughter, sentenced to two years imprisonment, suspended for two years, and ordered to pay £25,000 towards costs.

Jack was admitted to the Children's Assessment Unit of the Leicester Royal Infirmary Hospital around 10.15am. Dr Bawa-Garba ordered blood tests at 10.45am, but due to a "failure in the hospital's electronic computer system" she did not receive the blood test results until 4.15pm [50]. The defence argued that the patient had passed the point of no return at some stage even earlier than 4pm.

The case involves many other factors, which have raised wide concern [51].

3.8 Radiotherapy and spreadsheets

We have defined bugs as when the computer program fails to correctly implement its requirements. Bugs therefore arise when a computer system is "improved" and the original requirements are not updated to reflect those changes. The computer system thereby no longer correctly implements the requirements in use. We may argue whether the bug is better understood to be in the computer or in the original requirements, which are now obsolete: but the definition of a bug is that there is a failure of the requirements to be correctly implemented. The point is, the system is not doing what its users reasonably expect, and there is a bug however we might want to explain it away.

When a new computer system was introduced in 1982 and until 1991 just under 1,000 patients who underwent radiotherapy at the North Staffordshire Royal Infirmary received a dose of radiation significantly less than intended [52]. The new computer system had changed the way radiotherapy calculations were made, but radiographers assumed an earlier manual adjustment still needed making. In 2006, an analogous error was made at the Beatson Oncology Centre, Glasgow, but in the other direction, increasing the dose, and with fatal consequences.

Lisa Norris received a dose of radiation much greater than that intended. Again, a change in software was made, but the implications were not evaluated. Her case has been thoroughly investigated [52], from which we quote extracts below. Noteworthy is that the investigation did not think there was any problem with the computer system, and they conclude blame can be attributed to Principal Planner A (the name is anonymised in the report).

> 2.7 This report makes frequent reference to the computer systems used at the BOC [Beatson Oncology Centre] for treatment planning. Particular reference is made to Varis 7, Eclipse and RTChart (registered trade marks). In this regard, it should be noted that at no point in the

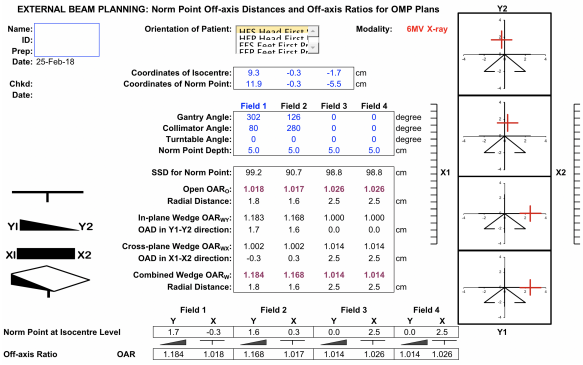

Figure 3: **Part of an Excel spreadsheet (with identifying information removed) for performing radiotherapy calculations. This spreadsheet has 19,218 cells including some 5,364 numbers. There is no evidence that the data in the cells is sanity-checked for any properties (e.g., that numbers fall within safe ranges).**

> investigation was it deemed necessary to discuss the incident with the suppliers of this equipment since there was no suggestion that these products contributed to the error.
>
> 5.28 A further critical circumstance was that the written procedures that were available to Planner B [. . .] the latest available version of this document (Annex 1 to this report), is dated 11th August 1998. . . .
>
> 6. Changing to the new Varis 7 system introduced a specific feature that, if selected by the treatment planner, changed the nature of the data in the Eclipse Treatment Plan Report relative to that in similar reports prior to the May 2005 upgrade. This feature was selected but the critical error was that the treatment planner who transcribed the resulting data from the Treatment Plan Report to the paper form (the planning form) was unaware of this difference and therefore failed to take the action necessary to accommodate the changed data.
>
> 12. It is important to note that the error described above was procedural and was not associated in any way with faults or deficiencies in the Varis 7 computer system.
>
> 16. The report concludes that most of the responsibility and hence any blame that can be attributed to treatment planning staff at the BOC falls to the staff member referred to in the report as Principal Planner A.

Principal Planner A is blamed when the Varis 7 software has a bug: it does not follow the documented procedures. Who was responsible for avoiding or fixing that bug is not explored in the report — either the documentation should have been updated so Varis 7 did was what expected, or Varis 7 should not have done unexpected things. As is clear from the previous experience at the North Staffordshire Royal Infirmary, installing new software that has hidden surprises is a common problem. It is as if people think computers, and especially new and updated computers, are always good [53] — and in the case of Lisa Norris, the investigators do not even think it worth exploring what went wrong with the software or the procurement.

A telling comment in the report is

> 15.v. The potential improvements to patient safety following the introduction of new technologies were not properly assessed or implemented.

It appears, insofar as anything can be concluded from a single sentence, that the assumption is that computers improve patient safety, so if an error occurs after such an "improvement," it is logical to blame the user.

Figure 3 illustrates a typical Excel spreadsheet as used for radiotherapy calculations. This particular spreadsheet has over 19,000 cells of data.

Consider the following tiny Excel spreadsheet with just 3 cells:

Row	Cell contents	Column A
1		1.00
2		2.00
3	`=SUM(A1:A2)`	1.00

The Excel `SUM` calculation is clearly wrong; the total of the two cells, A1 and A2, should be $1.00 + 2.00 = 3.00$ — *not* 1.00. Note that the error in the spreadsheet is only so obvious because this is a tiny spreadsheet constructed to clearly show the problem; in a spreadsheet used for a real problem (such as the one exhibited in figure 3) the chances of spotting such an error are negligible — and every time the spreadsheet is used, it needs to be checked again in case some unnoticed typo has changed a cell to an incorrect value.

The reason is that cell A2 contains ="2.00", a string value, rather than the numerical value it looks like. The function `SUM` treats any cell that is not a number as zero, so even though it looks like 2.00, Excel treats it as zero for the sum. Unfortunately, Excel does not warn the user that the contents of cell A2 are ignored. Of course, this example is a deliberate bug constructed to make a point in this paper, but with a radiotherapy spreadsheet of over 19,000 cells, one wonders what accidental and unknown bugs would lie unnoticed.

Microsoft are in an interesting position: if they recognise this problem as a bug (and we have been widely publishing the problem since 2010 [39, 54, 55]), then fixing it will make many existing spreadsheets fail, or worse it may make existing spreadsheets calculate different answers, which would be a different sort of problem. On the other hand, Microsoft could add a feature to detect and report on such problems, so that spreadsheet users can recognise when their spreadsheets are potentially unreliable — surely, this is the very minimum that any healthcare spreadsheet user should require? Indeed, a cautious spreadsheet user, especially a user of a large spreadsheet, should include many checks to help confirm all data in the spreadsheet is reasonable [56] — otherwise, one day, an accidental slip could change valid cell data into an error, with repercussions for patient treatment. (The reference [56] also contains other useful recommendations such as pair programming — having two people check each other's work for errors they may not notice if working on their own — when programming spreadsheets.)

3.9 Less obvious bugs

The common view of a bug is that the computer or program suddenly stops working; it crashes. More often poor design causes frustration, inefficiency and even cover-ups. Examples from healthcare include:

Lack of interoperability. Computer systems are developed independently and do not work together. Patient data cannot be transferred from one system to another.

Over-zealous and inappropriate security. Clinicians have to enter multiple passwords to use computers. Their work is considerably slowed down by such non-clinical workload.

Workarounds. Computer systems impose constraints on what clinicians can do (or what they can report they are doing), so clinicians effectively lie to the computers just to get things done.

Lack of validation. Users may, for all sorts of reasons, make errors entering data but a well-designed computer system will validate their data. If a user enters a number without the decimal point, the drug dose may be ten or a hundred times too high, but as there are generally recommended drug doses, this error can be trapped and the user warned, and either forbidden from entering such an extreme value (hard validation), or asked to confirm that is really what they require

(soft validation). It is very hard to design good validation: programmers make mistakes just as users do, so they may overlook opportunities for validation, and their validation (if done) may itself be buggy; and what seems like useful validation to a programmer may be ineffective for the user.

For example, Mersey Burns, which is an award-winning medical app, will soft warn the user that an entered weight of 4,880 kg is high for a newborn, but if the user thinks they entered 4.880 kg, the warning will most likely be lost on them — and if the user tries twice to "make sure," the validation warning does not repeat! Worse, over-zealous validation may force users to try workarounds — one real example was a nurse who changed the patient weight in order to deliver a correct drug dose (checked against the weight), but then forgot to correct the weight afterwards.

Ambiguity. Drug names may be too long to display in full, and may be misread. Patient records may be too long to fit on one screen, so a clinician may be unaware of further text (drug allergies?) on subsequent pages.

Obfuscation. User interfaces may "work" but they may be unnecessarily complicated, or complicated in unknown or unexpected ways. This will cause problems for staff and patients that typically manufacturers will deny are their fault.

Configuration errors. Computers may work, but hospitals may configure them and make mistakes in their configuration. A common example is setting up drug libraries (which provide critical information on drugs used in each hospital): mistakes here — bugs — are not in the software but in the local data. Another example is for third parties (e.g., service engineers) to reconfigure a system and cause problems — this is equivalent to an internal cybersecurity breach.

Big bang. It is very hard to gradually install a new system, so most computer systems are installed with a "big bang." One day, everything is new and different (and nobody knows how to use it) — it will likely have teething problems, that is, bugs that were not anticipated.

3.10 Invisible and denied bugs

Hospitals do not know what bugs their systems have. The economic consequence is that suppliers sell features — i.e., benefits that are obvious — rather than dependability or safety, since nobody knows how to measure that reliably. Curiously, we know how to measure the cost of patient harm and litigation, but we have no idea how to measure the benefits of computers, let alone whether the benefits offset the costs! Not only do we have little idea how to assess (or regulate) the quality of hospital computers; we have no idea how to quantify the relation between computers, their quality and patient outcomes, so it is not obvious how to improve.

In contrast to hospital computer systems, many goods, such as electrical items and pharmaceuticals are regulated so that they have to be appropriately safe. Yet the marketing story presented to healthcare is that computers are fantastic. So, if we believe the hype, when a problem occurs — as one eventually will — the cause can only be the users, generally the front-line clinicians. This impeccable logic is also backed up by our legal culture: many systems are provided on contractual terms that hold the manufacturers harmless from liability, yet the law also indicates that computers are reliable [57, 58]. Many systems contractually claim that clinical judgement is the final responsibility — despite the obvious fact that many computers are used *because* they perform clinical judgements (calculators would be a very simple example — they are used because they tell the professional what to do). In the EU, medical device regulation essentially specifies that if a device has a CE mark, and did not malfunction (and if it did, that is probably claimed as a hospital maintenance problem), then the manufacturer is protected.

When the Princess of Wales Hospital identified some problems with nursing, it resulted in over 70 nurses being disciplined and five indicted in court. This, of course, was at a tremendous cost to the hospital, as well as to patients and to the hospital's public image. Some nurses pleaded guilty and had custodial sentences. However, some nurses pleaded not guilty, and we were involved in the court case as expert witnesses, and hence had access to the relevant data and statements from the manufacturers. The manufacturer's opening evidence said that the systems were CE marked, and therefore any problems were the nurses' fault. The case has been described in [59], summarised in the legal reference book *Electronic Evidence* [60], and the judge has published his final ruling [61], which led to the collapse of the court case. In summary, the computer data was corrupt but nobody had noticed. Instead, the police had pursued confidently blaming many nurses for the problems.

In hindsight, it hardly seems insightful to point out that 70 corrupt nurses is far less plausible than a single corrupt database. Computers fail all the time, as the prosecution admitted in court. Nurses very rarely "fail," and when they seem to it is more likely because they are working in onerous conditions with unreliable computer systems.

It is sobering to wonder what other hospital data may be corrupt (whether from bugs, inadequate data management, or cyberattacks, etc) but is being misdiagnosed. The answer is we simply do not know. And blaming nurses is a recipe for not finding out.

3.11 Everyday examples

The problems we listed above were chosen because they were easy to explain, and they may therefore seem easy to dismiss because "obviously" clinicians should not be so unprofessional as to make the "simple" errors we discussed. This response ignores the complexity and pressures clinicians work under: errors of the sort we showed are routine to make because of the highly-pressurised clinical environment.

The response also comes out of a misunderstanding of human error. We don't think we make many if any errors, so other people making errors must be incompetent. The truth is, we do make errors — we make errors all the time that we do not notice. If we noticed our errors, of course we would not make them! Add up ten numbers with a calculator, then do it again in the reverse order — you will often get different answers. But each time you added up the numbers you did not know you had made an error, until you made the final comparison. In other words, we make errors more often than we think — and so do other people. The "unlikely" errors we discussed above are far more likely than we think.

Another natural response might be that the problems should be mitigated by proper staff training. Clinicians are supposed to be professionals after all. Unfortunately, nobody notices the problems we are discussing here, so training to avoid problems is unavailable. More practically, there are thousands of clinicians (the NHS is the UK's largest employer) and a training programme for them would not be able to keep up with technical innovation.

Rather, we should see the problems as symptoms of a worrying underlying new disease of epidemic proportions. And once we recognise the symptoms of this disease, poor programming becomes visible everywhere. If we avoided the bugs through better software engineering and regulation, then everything would be safer.

A calculator user may make an error that they notice and therefore wish to correct. The Casio HR-150TEC, like many calculators, has a delete key to help correct errors. Unfortunately, on the HR-150TEC the delete key ignores any decimal point. Hence trying to correct entering 2•5 to the intended 25 will likely leave 5, a result that is immediately out by a factor of five.[3] Decimal points are hard

[3] If the user keys 2•⟨DELETE⟩5, they will get 5; if the user keys 2•5⟨DELETE⟩⟨DELETE⟩5 they will get 5.

to see [62], and if the number is involved in a longer calculation, the final error may be very hard to spot.

The HR-150TEC can keep a record of what it does. The HR-150TEC's log is on paper, but many medical devices record internal data logs that are not visible. If a user encounters the design error described above, the log records what the calculator did not what it was told to do. If used in an investigation, then, the logs would seem to show the user erroneously entered 5 but did not correct it. In fact, the error is made by the calculator and, worse, the error is in a feature marketed to help the user correct errors!

Like Casio, Apple are market leaders and represent industry practice. The Apple iPhone calculator will be easily available to many. In Version 11.2.5 (the current version as of January 2018) of the Apple operating system, pressing AC ± ⟨SWIPE⟩[4] ± in the Apple calculator will result in Error being displayed, meaning that something unexpected has happened. That is, the Apple programmers have detected an error in their program code, where the program cannot handle this basic sequence of user input. It is a bug. This particular bug fortunately says Error but it is a silent warning and the user may nevertheless continue with their calculation, and then they will get the wrong result, unaware that an error occurred part way through their calculation. It is easy to create examples where the final answer is wrong, but Error is not displayed. Such examples may seem contrived, but they should not happen at all. (Technically, what we showed is a MWE, a minimum working example — if we showed more a realistic example, the nature of the bug would be harder to see so clearly.)

But the examples raise the serious question: how much else is badly implemented in calculators? The fact that such simple examples are programmed incorrectly suggests that there is a widespread quality control problem that may affect any calculation. Consider how many calculators have been sold and are in use: even though the probability of a bug affecting a user may be quite low, the number of users times the probability is worryingly large. Apple and Casio are by no means unique: almost every calculator has similar peculiar bugs that can catch users out [55].

⚠ The next few paragraphs illustrate that discussing bugs in enough detail to understand them is, frankly, tedious, even though people can get killed by them. Tracking down, diagnosing and fixing bugs is not exciting. If you were a nurse, every day you would have to work through many calculations like those in the example below, and as in the case study, you would face the unnecessary risks of poor programming in the systems you use.

Consider a realistic healthcare calculation, which illustrates how use errors are exacerbated by bugs. The pharmacy gives a nurse a drug bag, which is labelled that it has 130 mL of drug at a concentration of 45.57 mg/mL, and that the patient should be dosed at 5250 mg over 4 days. The nurse will work out the rate to set the infusion pump to, to deliver the drug at the right rate to the patient. This isn't particularly easy, and one wonders why the hospital pharmacy computers did not do the calculation to save the nurse doing it: they could have made this part of the nurse's job easy — and reliable. Instead, the nurse has to do the calculation 5250/45.57, which gives a dose in millilitres per day. But the infusion pump must be set in millilitres per hour, so this needs dividing by 24 hours in a day. Put in the language of a calculator: the nurse must press AC 5250÷4÷45•45÷24 =, and the result should be 1.2 mL per hour.

Two nurses working in a team performed this calculation. Unfortunately both worked out an answer 24 times too high, and they therefore agreed on it, but the high dose was fatal [63].

We do not know exactly how the error occurred. Perhaps one nurse forgot the 24, and the other missed out the last / in the calculation (or typed • or something else instead of it) — most calculators allow a user to miss out a divisor and divide by 1 instead of reporting an error. If so, the nurses would have got 28.8779 and 28.8764, respectively, which are the same to 4 figures. Or they may coincidentally have

[4] Swiping your finger across the display will normally delete a digit (or decimal point) from the display. It is a useful way to correct errors.

made exactly the same errors and got exactly the same answers? For systems (calculators, infusion pumps) in a hospital, to ignore and unhelpfully disguise a use error is a bug.

We have designed a drug dose calculator that detects *and blocks* these errors [64], though it would be even better built into the infusion pump (and other devices) to avoid the final error-prone step of the nurse copying a number from the calculator's display to the infusion pump's buttons. Perhaps the pharmacy or the drug bag itself should program the infusion pump directly?

In another tragic example, again where too few details are known (why don't devices keep proper logs?), the nurse, after making a calculation error of some sort, committed suicide [65, 66].

Why do nurses have to do what computers are — or could easily be — best at? If calculator manufacturers with all their computing and programmings skill cannot get basic things right, it is unlikely that medical device manufacturers will be better.

3.12 Confidentiality and "cargo cult" research doesn't help

We have mentioned bugs in specific systems with care, which are either well-documented or can be reproduced by the readers as we describe them; almost any systems could have been chosen for our examples. We selected our examples above because they are simple enough to explain clearly in this paper, and because they are representative of widespread problems (though we do not pretend this paper is a systematic review; indeed one systematic review concludes, as we do, that there is a paucity of reliable research [67].

There is in fact no shortage of more complex examples, such as serious problems with radiotherapy systems. We mentioned the 1985–1987 problems with the Therac 25 earlier (see section 2.3), and then over the period 2000–2001, the radiotherapist Olivia Saldaña González treated patients who were overdosed and died. She became a martyr to the cause: imprisoned for manslaughter caused by bugs [68, 69]. Many other examples are discussed in [42, 63].

We named our examples. We are worried by the alternative approach where products are anonymised, which limits what people can learn, and seriously limits manufacturers' and hospitals' ability to improve safety.

The paper [16] (which is by no means unique, but we cite it by way of giving a concrete example; we also cited it on page 6) compares two hospital systems (specifically EHRs, electronic health record systems), called in the paper *EHR1* and *EHR2*. One is three times worse than the other, at least for the experimental task it was evaluated with. Surely it is not only normal scientific practice but imperative to know the identities of the systems?

- We don't know what systems were tested;
- the research does not help the manufacturers improve their products;
- the research does not help future science replicate or extend it — in this case, this is critical because the paper does not clearly address *why* there were differences in safety;
- errors in the research, if any, cannot be corrected as nobody can check or replicate it — no data is available (not even the experimental task);
- the research does not help anyone who wants to buy safer systems;
- the research does not help patients who want safer treatment or staff who want to work in safer hospitals;
- do we want patients just to worry they are being treated with the worse system?

In short (apart from gaining a publication for the authors) the paper fails to empower anyone — and preventable bugs in hospital computer systems will remain despite it. Within the confidential culture of healthcare computing (discussed below), such papers may be the best we can do, but they are, nevertheless, examples of so-called cargo cult research: they look like research, but fail the accepted tests of good science [70].

4 Pathways to improvement

We have shown there are ubiquitous problems, and that there are likely adverse consequences that could be avoided. For example, we use calculators because we do not know the answers, so there ought to be an obligation to make them reliable. Why aren't computers more reliable, especially in healthcare?

- It is politically expedient that newer IT will solve problems automatically: hospitals do not have the latest IT, and it is more profitable to market "new technology" rather than to solve the underlying problems.
- Everybody thinks programming is easy (and calculators — an example above — are trivial as things go), so no adequately professional effort goes into their design.
- It is easier to blame users for error than to blame designs (which are protected in law), and errors occur because users are unaware of them. Error is hard to replicate and users involved in error often want to conceal it. In contrast, cancer, say, is not the hospital's nor clinician's fault and there are recognised approaches to understand it. Solutions to cancer will doubtless make money for industry, but solutions to preventable harm will save money for the NHS.
- There is understandable resistance from industry to undertake research that may lead to improved safety standards, a more cautious purchasing regime, or any increase in the so-called "regulatory burden."
- Koppel *et al* [71] exposes the standard confidentiality and "hold harmless" legal arrangements that stifle research, and in particular, reproducible research — there are studies of systems A and B, but if we don't know what they are (for "legal reasons") then nothing can be learned, other than that there is variation in quality of systems. We gave a recent example in section 3.12, above.

When it seems that a clinician has made a mistake, why don't we first ask: did a bug contribute to that outcome? It probably did. (Big bugs like crashes may be obvious, but there are also hard-to-see bugs, when a program to fails to work correctly but sort-of works.) Section 3.10 discussed a case where over 70 nurses were disciplined and some indicted and prosecuted in court for making errors we showed were actually caused by bugs (and mismanagement) that had been overlooked [59].

4.1 14 suggestions

We ought to improve safety. Here are a few suggestions:

1. If it is impossible to tell whether or not a computer contributed to an error, presume it did for legal purposes. (The current presumption in UK law is that computers are reliable [57,60,72].) If we required this, very soon manufacturers would put effective black boxes into their products. Moreover, the black boxes should use open standards — currently, only the manufacturer can fully interpret data yet there is a conflict of interest because they bear liability if it turns out the data implies their product failed [73].

2. Every device and system must be safety tested and rated. We have discussed elsewhere how to do this, e.g., [34,35]. The ratings could then be used to label the devices, like we currently label white goods (like fridges) for energy efficiency as it encourages people to buy more efficient white goods, which in turn drives manufacturers to make them more efficient. Likewise, we should be driving manufacturers to make safer medical systems [74].

3. Design problems should be avoided or fixed as early as possible in the design cycle, before testing. Aviation software is developed to a higher standard than healthcare software. One aviation standard is DO-178C [75], showing that tighter standards are no obstacle to manufacturers. DO-178C could be adopted directly into healthcare, though to do so would need a transitional strategy, as what is routine in aviation (because people's lives depend on safe software!) is currently an insurmountably high hurdle for healthcare. But DO-178C shows it can be done; of course, a derivative standard might be developed to more closely align with clinical needs (e.g., including information governance). Other relevant standards include IEC 61508, 62304, ISO 15026, 14971, 9241, etc — there is no shortage of ways to improve, and we think DO-178C is just a start [76].

4. It seems self-evident to us that if clinicians have to be qualified and regulated before they can treat patients, then software engineers who develop and build safety-critical clinical systems should be qualified to at least comparable levels of competence, and so should the technicians who maintain these systems. Currently it takes about 8 years to qualify as an anæsthetist and it is a very responsible job, but you can start programming an infusion pump or patient record system this afternoon and have no relevant qualifications whatsoever. We note the recent founding of the Faculty of Clinical Informatics [77] and this, we hope, will help change the culture and support best practice.

5. Due to industrial pressure and consumer (also procurement) eagerness products are released as if they are finished products (sometimes they are released before they work). This is exacerbated because typical procurement contracts emphasise delivery rather than levels of performance. The ISO standard 9241 [33] shows that products must be iteratively designed: after delivery, they should be evaluated and continually improved. Many bugs and design defects will only become apparent after a system is used for real — hospitals are complex places, and many procedures are complex and probably not fully understood either by designers or procurement. A mismatch is inevitable. How does the manufacturer act on "post-market surveillance" and do they collect use data and user feedback? ISO 9241 provides a framework for improvement. Arguably, if ISO 9241 is not in procurement contracts, they are inadequate.

6. "Ordinary" consumer and office products like calculators and clinicians' own phones (and the software running on them) must be regulated if they are to be used in healthcare. Under current European legislation, they do not need regulation for use in healthcare.

7. So-called "hold harmless" and other warranty limitations for software must be banned [71]. Confidentiality is a standard clause, which further limits awareness of any problems. What incentive is there for making bug-free software if the manufacturers can deny liability for problems? Indeed, some software forces that the user agrees to indemnify the developers. For example, the Mersey Burns EULA (end user licence agreement) states that the developers are not liable for any damages, and furthermore that users indemnify the developers for any liability — this is not unusual. This is backwards and would be unacceptable for other products.

8. It is possible that fixing a safety vulnerability could be taken as an admission of prior liability. Laws need changing to encourage improvement, not to disincentivise it.

9. Designing, developing, procuring and monitoring safe systems is complex and error-prone. At all stages, oversight should be required [78]; explicit safety ratings (based on rigorous

evaluations, perhaps similar to safety ratings on other safety-critical products such as car tyres) would also help [63].

10. The regulatory process is very slow and ponderous, but technology is advancing much faster. This can create tensions and design compromises. Thus, oxygen is regulated as a pharmaceutical, but a computer regulating the flow of oxygen is regulated as a medical device; yet the two may be combined into a single, integrated computer-controlled oxygen cylinder. Pharmaceutical regulations (in the UK) forbid instructions or warnings for the integrated medical device controlling the oxygen to be placed on the cylinder — even though they are physically inseparable. Another example is that information governance was established in the 1960s, so a hospital can fax a patient (but does the right person get it?) — but they are typically forbidden from using WhatsApp or Skype, even though these are more secure and reliable ways of communicating. Therefore there must be a more balanced and more efficient way for technical innovation to inspire and nudge improved regulation, as well as regulation, in turn, to manage and reduce risk in innovation.
11. Leaders, thought leaders, and procurement need more training and awareness of the true capabilities and risks of computers, and of the processes that should be used in mature software engineering. Currently industry, understandably, wants to promote embracing new technology but this must be balanced against the evidence for the effectiveness of planned interventions. Healthcare needs a lot of wisdom to distinguish between the excitement of consuming new technology (which is valid for personal consumption) versus the real excitement of solving healthcare problems (which requires products to work reliably in complex healthcare environments) [53, 79]. Seeing computers as a medical intervention that should be subject to the same "evidenced based" core of healthcare thinking would be an improvement (and indeed would stimulate more, and more rigorous, research).
12. A deep cultural problem that needs to be addressed is that the clinical literature (e.g., on burn treatment) was never designed to help specify reliable software for clinical use. A very simple example is the clinical literature on body mass index does not specify that patients have a positive weight (they must weigh more than zero); a programmer implementing what the literature says will end up with software that will ignore use error such as entering a zero weight or perhaps not filling in a weight at all. Another example was given in our discussion of QRISK validation in section 3.5.
13. Cybersecurity research has exemption from confidentiality laws, such as commercial confidentiality law, otherwise inhibiting research [80]; it would be very easy to make safety another exemption.
14. Crucially, we need more research, and more quality, independent research. Although stories and examples are persuasive, we need rigorous science to discover the underlying principles of improvement that can help improve everything, and to get good evidence to prioritise action [81], political, regulatory and at the point of care.

This list is not exhaustive, but shows that a variety of approaches are feasible, many of which have been successfully tried-and-tested in other industries. Each idea, even in isolation, will help improve safety.

Legal, clinical, hospital and media culture is another matter. Thankfully, if we improve software design and programming, safety will improve regardless of healthcare culture. There are fewer developers to convert to a safety culture than clinicians, hospital managers and politicians; arguably, then, technical improvements — which aviation has path-broken — are the best way to start.

Note that the aviation standard DO-178C was developed by a volunteer organisation (the Radio Technical Commission for Aeronautics), so clearly developing standards need not wait for statutory sanc-

tion. And, of course, current healthcare software standards are widely recognised as being out of date (e.g., balancing technically-obsolete data protection adherence while permitting realistic, often innovative, clinical use of mashed up apps and servers in unspecified jurisdictions . . .), so they need updating anyway.

Adopting the ideas above certainly demands a change in perspective for developers. Hopefully, some will want to stand out as leading safety culture. And once culture change starts, it will stimulate more ideas beyond our initial list above.

4.2 Recognising there is a problem

The first step in solving a problem is to recognise there is a problem — and then to diagnose and treat it correctly. Even with serendipitous ideas, such as the discovery of penicillin, they are only recognised because they solve a known problem (in penicillin's case, of bacterial infection). In the case of bugs in hospital computers, few recognise there is a problem, therefore even fewer are researching the scope and reach of the issues, let alone the solutions.

The NHS does have many problems, and it is easy to jump to the conclusion that computers are the obvious solution to them, rather than first researching the actual benefits of computers for those problems. It is easy to confuse "new and exciting" for good [63, 82]. But when thinking about the NHS and new technology, we need to focus on actual effectiveness in supporting the NHS, its staff and patients — we need to do better than follow intuition.

The Times recently reported tragic cases of people dying after NHS postal paper mixups, and their article shows that to many it seems obvious that the NHS should use email more [83], to replace the old technology of paper with the new of email. Unfortunately, email itself doesn't cure the problems. The problems were due to human error, not to paper.

Email, in fact, often makes error worse and harder to see. Have you ever sent an email to the wrong person? Have you "replied to all" by mistake? In 2016, the NHS had to use, ironically, a recorded voice message after an email accidentally sent to 1.2 million people ground things to a halt — not least because some "replied all" to everyone causing an avalanche of emails. More recently, over two days (24–25 January 2018) NHS computer systems failed across Wales, with hospitals and GPs unable to access any information [84]; this nation-wide IT failure came at a time when the NHS was already facing unprecedented loads from staff shortages and winter flu.

More reliable systems could be developed to help detect, block and mitigate human error (face to face video helps; patients in charge of their records may help, AI can help). But until then, and when software has fewer bugs and failings in its design, email is just a faster way of doing everything, *including* making mistakes.

The conventional solutions involve buying more computers, because new computers are exciting and seem to offer huge transformational potential for healthcare; the UK Wachter Report [85] is a case in point expressing this view. Of course, there is a lot of commercial pressure driving the conventional solutions. The Engineering and Physical Sciences Research Council (EPSRC), which would seem to be the natural UK funder for the needed research, has a "healthcare technologies programme" [86], but it does not recognise the role for improving computing technology or the need for software engineering to be more reliable in delivering healthcare (or even in supporting more reliable research). The oversight is an international problem. One example is the only recent discovery of a 15 year old bug, the impact of which undermines thousands of papers and years of brain research [87]. The problem took so long to discover because nobody thought computer bugs were an issue.

At the most general level, when we recognise there is uncertainty or controversy in a critical area (such as healthcare), then strategic research to reduce that uncertainty is called for. Politicians and managers,

for instance, cannot make sensible plans about future computing in healthcare when even the broad impact of their decisions is unknown — past examples such as the UK National Programme for IT (a £11.4 billion programme scrapped in 2011) and very recent examples such as the IHealth Electronic Health Record System [88,89] give little reason for confidence in hope over science, let alone hope that we may have learned anything useful from past failures in healthcare computing projects. Consider that the Government predicted savings of £1.2 billion for "telehealth," but subsequent studies showed no real benefits, just increased cost [90].

Why is there this repeated hope in new solutions, when it seems to us that there are more fundamental problems that no new technology overcomes, however exciting. Where, given the history of disappointing results, is the unbiased research? What sort of solutions might be proposed and evaluated as priorities? We explore these questions in our conclusions, next.

5 Conclusions

Healthcare software bugs are based in a lack of awareness of software engineering best practice. They are preventable, and contribute to preventable harms. If the most technical people in the supply chain are unaware of bugs and how to avoid them, there is little hope that the end users — clinicians, regulators, patients — are adequately aware of bugs and how they undermine the delivery of safe healthcare. In particular, investigators are unaware how bugs undermine reliable analysis of incidents, and hence there is a tendency to blame the user and not correctly apportion system causes.

Most people know very little about programming, though they know that children can do it, so it must be very easy. Because most people — including those in healthcare — know little about programming it is very hard for them to recruit good programmers or to select quality suppliers; indeed, many medical device manufacturers cannot program themselves so they out-source all software development.

Most design consultancies and developers think they are good programmers. It is in fact very easy to write programs that look like they work. The errors we discussed above affecting B.Braun, Zimed and Apple software are cases in point. Similarly, it is easy for a cowboy builder to make things look like safe houses, but they conceal structural, electrical, and fire risks that are very hard to spot — until things go wrong. (It is easy for children to build houses in Lego … but it doesn't mean they are or are even going to be competent structural engineers.)

With our culture's unawareness of the hard skills needed for dependable programming, combined with many programmers' hubris, exacerbated by industry's continual pressure for healthcare to just buy solutions, it is going to be hard to get any of our suggestions above implemented.

WannaCry didn't make many say healthcare computer systems should improve — the emphasis was mainly on the NHS's responsibility to keep cyber-defences up to date. The costly suspension of over 70 nurses because of a computer error didn't make many say programming should improve. The deaths of patients and the imprisonment of Olivia Saldaña González didn't prompt many to say that programming quality or regulation should improve.

Research into safer use of computers will improve the quality of healthcare, save lives, and save unnecessary harms and the costs of litigation. It would save the NHS and all healthcare providers money and would improve the health of society — it is an international problem and solutions will have an international impact. In comparison, conventional clinical research is expensive and, when successful, generally results in drugs or devices that cost further money and only helps patients with specific diseases. Improving computers to avoid the bugs (and the regulatory and procurement cultures that acquiesce to them) as we discussed throughout this paper will save money and help *everybody*, both patients and staff, regardless of disease.

Cancer research	Computers in hospitals research
Expensive drug development and trials	Cheap program proving and evaluation
Each drug is different	Programming principles are general
Expensive treatment with drug	Negligible cost to benefit
Usually targets specific cancers	Benefits all patients
	Benefits staff
Usually has side-effects	No side effects
	Reduces negligence claims
May be complex to use	Improved usability
Usually patented	Improvements can be used by all manufacturers
Significant funding	*Negligible funding*

Figure 4: **An outline comparison of cancer research (to take one clinical example) and research into computer bugs.**

Consider three well-known accidents from outside of healthcare:

- The 1988 Piper Alpha disaster killed 167 people and led to a Public Inquiry [91], and to radical changes to the way safety is managed on off-shore oil platforms.
- The 1999 Ladbroke Grove train crash in London killed 31 people, and remains one of the worst rail accidents in Britain. A public inquiry into the crash was held in 2000 [92]. Interestingly, the crash would have been prevented by an automatic train protection (ATP) system — a computer system to *prevent* error — but it had previously been rejected because of cost.
- The 2017 Grenfell Tower fire tragedy killed 71 people and led to a Public Inquiry [93], as well as to a review [94] with plans for radical changes in the way that fire safety is regulated in high rise buildings.

These were major tragedies. It is right that they are leading to radical changes to reduce future risks.

For healthcare, the National Reporting and Learning System (NLRS) says there are about two million reported incidents per year in England and, of those, about 10,000 lead to serious harm or death. The review [3] suggests that NLRS detects only 7–15% of all incidents, so the true figure may be closer to 100,000 serious incidents, which agrees with our extrapolation from US figures [1] (see page 3).

If only 1% of that 100,000 (that is, 1,000) is computer-related, then each and every year computer-related serious harm or death exceeds the fatalities from major accidents like the Piper Alpha, Paddington, and Grenfell tragedies *combined*. This paper presented a wide variety of serious bugs, most of which are unnoticed, ignored or denied by stakeholders; we think the 1% guess for the computer-related rate is likely to be far too low, since computers are involved in every aspect of patient care.

Piper Alpha, Paddington, and Grenfell capture the imagination not just because they were tragically preventable but because they were visible. We hope that our paper exposes and makes clearly visible the far greater damage done every year by invisible (and denied) bugs in hospital computer systems, and why this hidden carnage need not continue.

What will it take before we get our Public Inquiry? When will we have the "radical changes" in healthcare regulations?

What will it take to prompt the needed funding for research to quantify, understand and solve the problems, and to overcome the current Catch 22 that "there is no reliable data to prove healthcare IT research is needed" — an excuse long familiar from the resistance to researching the facts of smoking.

Notes and further reading

This paper has reviewed computer bugs in healthcare and their impact — and noted the lack of research, whether to be certain of the scale of the problem, or to seek solutions. We recommend the following resources — we are not the only people with these concerns:

- Fu's Archimedes Center for Medical Device Security web site [95] has many resources on cybersecurity issues in medical devices and systems.
- The Royal Academy of Engineering is producing an authoritative report, *Cyber safety and resilience: strengthening the digital systems that support the modern economy*, which should be published in 2018.
- Koppel and Gordon's book, *First, Do Less Harm: Confronting the Inconvenient Problems of Patient Safety* [96] is an edited, multi-author book with further resources on computers and their impact on patient care.

We want our paper to inform and empower you, and we hope we have done both. We will look forward to hearing from you what you do.

Acknowledgements

The authors are grateful for very helpful comments from Carolyn Greig, Kevin Fu, Stephen Mason, Ross Koppel, Alford R. Taylor Jr. and Xixi Yao.

Funding

Harold Thimbleby is very grateful for support from See Change (M&RA-P), Scotland.

Declaration of Conflicting Interests

The Authors declare that there is no conflict of interest.

References

[1] James JT. A new, evidence-based estimate of patient harms associated with hospital care. Journal of Patient Safety. 2013;9(3):122–128.

[2] Department for Transport. Annual road fatalities; 2013. Available from: `www.gov.uk/government/publications/annual-road-fatalities`.

[3] Elliott RA, Camacho E, Campbell F, Jankovic D, St James MM, Kaltenthaler E, et al. Prevalence and economic burden of medication errors in the NHS in England. Rapid evidence synthesis and economic analysis of the prevalence and burden of medication error in the UK. Policy Research Unit in Economic Evaluation of Health & Care Interventions (EEPRU);

2018. Available from: www.eepru.org.uk/wp-content/uploads/2018/02/medication-error-report-revised-final.2-22022018.pdf.

[4] NHS Improvement. Commentary on National Patient Safety Incident Reports submitted to the National Reporting and Learning System April to June 2017; 2017. Available from: improvement.nhs.uk/uploads/documents/NaPSIR_commentary_Apr-Jun_2017_v3.pdf.

[5] Acharya C. Human-Computer Interaction and Patient Safety. Swansea University; 2017.

[6] National Audit Office. Managing the costs of clinical negligence in trusts; 2017. Available from: www.nao.org.uk/report/managing-the-costs-of-clinical-negligence-in-trusts.

[7] Ward V. More than 7,500 doctors warn they will be too scared to admit mistakes after pediatrician is struck off. The Telegraph. 2018;Available from: www.telegraph.co.uk/news/2018/01/28/7000-doctors-warn-medics-will-scared-admit-mistakes-pediatrician.

[8] NHS computer problems could be to blame for 'hundreds of deaths', academics claim. The Independent. 2018;Available from: http://www.independent.co.uk/news/health/nhs-computer-problems-blame-hundreds-deaths-harold-thimbleby-martyn-thomas-gresham-college-a8197986.html.

[9] Belluck P. First Digital Pill Approved to Worries About Biomedical 'Big Brother'. New York Times. 2017;Available from: www.nytimes.com/2017/11/13/health/digital-pill-fda.html.

[10] Institute of Medicine. Medical devices and the public's health: The FDA's 510(k) clearance process at 35 years. US National Academies of Science; 2011. Available from: www.nap.edu/read/13150/chapter/2.

[11] Reason JT. Human Error. Cambridge University Press; 1991.

[12] Magrabi F, Ong MS, Runciman W, Coiera E. An analysis of computer-related patient safety incidents to inform the development of a classification. Journal of the American Medical Informatics Association. 2010;17(6):663–670. Available from: www.ncbi.nlm.nih.gov/pmc/articles/PMC3000751.

[13] US Office of the National Coordinator for Health IT. HIT Policy Committee, Adoption/Certification Workgroup meeting; 2010. Available from: healthit.hhs.gov.

[14] Joint Commission on Accreditation of Healthcare Organisations. Sentinel Events; 2008. (accessed in in December 2008 but now missing). Available from: www.jointcommission.org/SentinelEvents/SentinelEventAlert/sea_42.htm.

[15] Han YY, Carcillo JA, Venkataram ST an, et al . Unexpected increased mortality after implementation of a commercially sold computerized physician order entry system. Pediatrics. 2005;116:1506–1512.

[16] Horsky J, Drucker EA, Ramelson HZ. Higher accuracy of complex medication reconciliation through improved design of electronic tools. Journal of the American Medical Informatics Association. 2017;0(0):1–11.

[17] Sinsky C, Colligan L, Li L, Prgomet M, Reynolds S, Goeders L, et al. Allocation of Physician Time in Ambulatory Practice: A Time and Motion Study in 4 Specialties. Annals of Internal Medicine. 2016;165:753–760.

[18] Koppel R, Lehmann C. Implications of an emerging EHR monoculture for hospitals and healthcare systems. Journal of the American Medical Informatics Association. 2015;22:465–471.

[19] MacAskill E. Destructive attack on UK a matter of 'when, not if', warns cyber chief. The Guardian. 23 January, 2018;p. 1 & 12.

[20] Koppel RJ, Thimbleby H. Lessons From the 100 Nation Ransomware Attack; 2017. Available from: thehealthcareblog.com/blog/2017/05/14/lessons-from-the-100-nation-ransomware-attack.

[21] Fu K, Thimbleby H. Ransomware: How we can climb out of this mess?; 2017. Available from: www.healthcareitnews.com/blog/ransomware%E2%80%A8-how-we-can-climb-out-mess.

[22] National Audit Office. Investigation: WannaCry cyber attack and the NHS; 2017. Available from: www.nao.org.uk/report/investigation-WannaCry-cyber-attack-and-the-nhs.

[23] Hern A. Hacking risk leads to recall of 500,000 pacemakers due to patient death fears. The Guardian. 2017;Available from: www.theguardian.com/technology/2017/aug/31/hacking-risk-recall-pacemakers-patient-death-fears-fda-firmware-update.

[24] Hoyme K. Ruminations on challenges in securing medical devices; 2017. Available from: www.fda.gov/downloads/MedicalDevices/NewsEvents/WorkshopsConferences/UCM559600.pdf.

[25] FDA. The FDA's role in medical device cybersecurity; 2017. Available from: www.fda.gov/downloads/MedicalDevices/DigitalHealth/UCM544684.pdf.

[26] Leveson N. In: Medical Devices: The Therac 25. Addison-Wesley; 1995. Available from: sunnyday.mit.edu/papers/therac.pdf.

[27] Molteni M. Medicine is going digital. The FDA is racing to catch up. Wired. 2017;Available from: www.wired.com/2017/05/medicine-going-digital-fda-racing-catch.

[28] Science and Technology Committee. Oral evidence: Algorithms in decision-making, HC 351; 2018. Available from: data.parliament.uk/writtenevidence/committeeevidence.svc/evidencedocument/science-and-technology-committee/algorithms-in-decisionmaking/oral/76942.pdf.

[29] Thomas M. Making software correct by construction. Gresham College; 2017. Available from: www.gresham.ac.uk/lectures-and-events/making-software-correct-by-construction.

[30] Christensen C, Grossman JH, Hwang J. The Innovator's Prescription. McGraw-Hill; 2008.

[31] Stead WW, Lin HS, editors. Computational Technology for Effective Health Care: Immediate Steps and Strategic Directions. National Research Councils; 2009.

[32] Linn MC, *et al*. Report of a Workshop on The Scope and Nature of Computational Thinking. National Academies Press; 2010.

[33] ISO 9241-210:2010. International Organization for Standardization; 2010.

[34] Thimbleby H. Contributing to Safety and Due Diligence in Safety-critical Interactive Systems Development. In: Proceedings ACM SIGCHI Symposium on Engineering Interactive Computing Systems — EICS'09. ACM; 2009. p. 221–230.

[35] Masci P, Rukšėnas R, Oladimeji P, Cauchi A, Gimblett A, Li Y, et al. The Benefits of Formalising Design Guidelines: A Case Study on the Predictability of Drug Infusion Pumps. Innovations in Systems and Software Engineering. 2015;11(2):73–93.

[36] Harrison MD, Masci P, Campos JC, Curzon P. Verification of user interface software in PVS: the example of FDA requirements and programmable medical devices. IEEE Transactions on Human Machine Systems. 2017;47(6):834–846.

[37] Cox AL, Oladimeji P, Thimbleby H. A Performance Review of Number Entry Interfaces. In: Proceedings IFIP Conference on Human-Computer Interaction. vol. 8117 of Lecture Notes in Computer Science. Springer; 2013. p. 365–382.

[38] Cox AL, Oladimeji P, Thimbleby H. Number Entry Interfaces and Their Effects on Error Detection. In: Proceedings IFIP Conference on Human-Computer Interaction — Interact 2011. vol. 6949 of Lecture Notes in Computer Science. Springer; 2011. p. 178–185.

[39] Thimbleby H, Cairns P. Interactive numerals. Royal Society Open Science. 2017;4(160903).

[40] Cauchi A, Gimblett A, Curzon P, Masci P, Thimbleby H. Safer title=5-key Number Entry User Interfaces using Differential Formal Analysis. In: Proceedings BCS Conference on Human-Computer Interaction. vol. XXVI; 2012. p. 29–38. Available from: dl.acm.org/citation.cfm?id=2377916.2377921.

[41] Norman DA. Design Rules Based on Analyses of Human Error. Communications of the ACM. 1983;26(5):254–258.

[42] Schiff G, *et al* . Computerized Prescriber Order Entry Medication Safety (CPOEMS): Uncovering and learning from issues and errors; 2017. Available from: www.fda.gov/downloads/Drugs/DrugSafety/MedicationErrors/UCM477419.pdf.

[43] Institute for Safe Medication Practices Canada (ISMP Canada). United States Marshals Seize Defective Infusion Pumps Made by Alaris Products. Kansas City infoZine. 2006;Available from: www.infozine.com/news/stories/op/storiesView/sid/17358/.

[44] ALERT: Potential for Key Bounce with Infusion Pumps. ISMP Canada Safety Bulletin. 2006;6(6).

[45] Iacobucci G. Computer error may have led to incorrect prescribing of statins to thousands of patients. British Medical Journal. 2016;353(i2742).

[46] Heather B. QRisk2 in TPP fixed but up to 270,000 patients affected. DigitalHealth; 2016. Available from: www.digitalhealth.net/2016/06/qrisk2-in-tpp-fixed-but-up-to-270000-patients-affected/.

[47] NHS England. NHS England Clinical Guidance: QRISK2 Incident; 2016. Available from: www.tpp-uk.com/wp-content/uploads/2016/06/NHS-England-Clinical-Guidance-QRISK2-Incident.pdf.

[48] Hippisley-Cox J, Coupland C, Brindle P. Development and validation of QRISK3 risk prediction algorithms to estimate future risk of cardiovascular disease: prospective cohort study. British Medical Journal. 2017;357(j2099).

[49] Daily Mail Reporter. Mother-of-four dies after blundering nurse administers TEN times drug overdose. Daily Mail. 2011;Available from: www.dailymail.co.uk/health/article-1359778/Mother-dies-nurse-administers-TEN-times-prescribed-drug.html#ixzz56FSSFvBQ.

[50] Leveson B, Openshaw, Males. Between: HADIZA BAWA-GARBA Appellant and THE QUEEN. Royal Courts of Justice; 2016.

[51] Dabydeen L, Klonin H, Wariyar S, Speight N, Holt K. An account by concerned UK paediatric consultants of the tragic events surrounding the GMC action against Dr Bawa-Garba. 54000doctorsorg. 2018;Available from: 54000doctors.org/blogs/an-account-by-concerned-

uk-paediatric-consultants-of-the-tragic-events-surrounding-the-gmc-action-against-dr-bawa-garba.html.

[52] Johnston AM. Unintended overexposure of patient Lisa Norris during radiotherapy treatment at the Beatson Oncology Centre, Glasgow in January 2006. Scottish Executive; 2006.

[53] Thimbleby H. Trust Me, I'm A Computer. Future Healthcare Journal. 2017;4(2):105–108.

[54] Thimbleby H, Cairns P. Reducing Number Entry Errors: Solving a Widespread, Serious Problem. Journal Royal Society Interface. 2010;7(51):1429–1439.

[55] Thimbleby H. Safer User Interfaces: A Case Study in Improving Number Entry. IEEE Transactions on Software Engineering. 2015;41(7):711–729.

[56] Bewig PL. How do you know your spreadsheet is right? Computing Research Repository. 2013;abs/1301.5878. Available from: arxiv.org/abs/1301.5878.

[57] Mason S, Seng D, editors. Electronic Evidence. 4th ed.; 2017. Available from: ials.sas.ac.uk/digital/humanities-digital-library/observing-law-ials-open-book-service-law/electronic-evidence.

[58] Mason S. Artificial intelligence: Oh really? And why judges and lawyers are central to the way we live now — but they don't know it. Computer and Telecommunications Law Review. 2017;23(8):221–222. Available from: stephenmason.co.uk/wp-content/uploads/2017/12/Pages-from-2017_23_CTLR_issue_8_PrintNEWMASON.pdf.

[59] Thimbleby H. Cybersecurity problems in a typical hospital (and probably all of them). In: Parsons M, Kelly T, editors. Developing Safe Systems: Proceedings of the 25th Safety-Critical Systems Symposium. Centre for Software Reliability; 2017. p. 415–439.

[60] In: Mason S, Seng D, editors. Analysis of a failure. 4th ed.; 2017. p. Section 9.90:321–323. Available from: ials.sas.ac.uk/digital/humanities-digital-library/observing-law-ials-open-book-service-law/electronic-evidence.

[61] Crowther T. CASE RULING: ENGLAND & WALES, Case citation: R title=v Cahill; R title=v Pugh, Ruling 14 October 2014. Digital Evidence and Electronic Signature Law Review. 2017;14:67–71. Available from: journals.sas.ac.uk/deeslr/article/view/2541/2499.

[62] Thimbleby H. Reasons to Question Seven Segment Displays. In: Proceedings ACM Conference on Computer-Human Interaction — CHI 2013. ACM; 2013. p. 1431–1440.

[63] Thimbleby H, Lewis A, Williams JG. Making Healthcare Safer by Understanding, Designing and Buying Better. Clinical Medicine. 2015;15(3):258–262.

[64] Thimbleby H. Ignorance of Interaction Programming is Killing People. ACM Interactions. 2008;15(5):52–57.

[65] Saavedra SM. Remembering Kimberly Hiatt: A Casualty of Second Victim Syndrome; 2015. Available from: nurseslabs.com/remembering-kimberly-hiatt-casualty-second-victim-syndrome/.

[66] Woodward S. Rethinking Patient Safety. Productivity Press; 2017.

[67] Ellsworth MA, Dziadzko M, O'Horo JC, Farrell AM, Zhang J, Herasevich V. An appraisal of published usability evaluations of electronic health records via systematic review. Journal of the American Medical Informatics Association. 2017;24(1):218–226.

[68] Borrás C. Overexposure of radiation therapy patients in Panama: problem recognition and follow-up measures. Pan American Journal of Public Health. 2006;20((2/3)):173–87.

[69] McCormick J. We did nothing wrong: Why software quality matters. Baseline. 2004;Available from: www.baselinemag.com/c/a/Projects-Processes/We-Did-Nothing-Wrong.

[70] Feynman RP. Cargo Cult Science, 1974 CalTech commencement address. Feynman RP, Leighton R, editors. Vintage; 1992.

[71] Koppel RJ, Kreda D. Health care information technology vendors' *hold harmless*, clause: implications for patients and clinicians. Journal of the American Medical Association. 2009;301(12):1276–1278.

[72] Mason S. Electronic evidence: A proposal to reform the presumption of reliability and hearsay. Computer Law & Security Review. 2014;30:80–84.

[73] Garber AM. Modernizing device regulation. New England Journal of Medicine. 2010;362(13):1161–1163.

[74] Thimbleby H. Improve IT, improve health. IEEE Computer. 2017;50(6):86–91.

[75] Rierson L. Developing safety-critical software: A practical guide for aviation software and DO-178C compliance. CRC Press; 2013.

[76] Thomas M. Safety Critical Systems. Gresham College; 2017. Available from: www.gresham.ac.uk/lectures-and-events/safety-critical-systems.

[77] Faculty of Clinical Informatics; 2018. Available from: www.facultyofclinicalinformatics.org.uk.

[78] Shneiderman B. The dangers of faulty, biased or malicious algorithms requires independent oversight. Proceedings National Academy of Sciences. 2016;113(48):1358–13540.

[79] Thimbleby H, Koppel R. The Healthtech Declaration. IEEE Security & Privacy. 2015;13(6):82–84.

[80] Alva A. DMCA security research exemption for consumer devices. Federal Trade Commission; 2016. Accessed 1 November 2017. Available from: www.ftc.gov/news-events/blogs/techftc/2016/10/dmca-security-research-exemption-consumer-devices.

[81] CHIMED team. Manifesto for medical devices; 2017. Accessed 1 November 2017. Available from: www.chi-med.ac.uk/publicdocs/WP345.pdf.

[82] Kahneman D. Thinking, Fast and Slow. Penguin; 2012.

[83] Lay K. Paper records put patient lives at risk. The Times. 23 January 2018;p. 6.

[84] Hodgson S, Evans O. Reports of widespread Welsh NHS system failure affecting hospitals and GPs. Daily Post. 2018;Available from: www.dailypost.co.uk/news/north-wales-news/live-welsh-nhs-system-failure-14197931.

[85] Wachter RM, editor. Making IT Work: Harnessing the Power of Health Information Technology to Improve Care in England. Report of the National Advisory Group on Health Information Technology in England; 2016. Available from: www.gov.uk/government/uploads/system/uploads/attachment_data/file/550866/Wachter_Review_Accessible.pdf.

[86] Engineering and Physical Sciences Research Council. EPSRC Healthcare Technologies Strategy Summary; 2015. Available from: www.epsrc.ac.uk/files/research/htstrategysummary.

[87] Eklund A, Nichols E Thomas, Knutsson H. Cluster failure: Why fMRI inferences for spatial extent have inflated false-positive rates. In: Proceedings National Academy of Sciences. vol. 113(28); 2016. p. 7900–7905.

[88] Cochrane D. Review of the Functioning of IHealth: Nanaimo Regional General Hospital. Oceanside Health Centre and Dufferin Place; 2016. Available from: ihealth.islandhealth.ca/wp-content/uploads/2016/11/ihealth-review-2017.pdf.

[89] Ernst & Young, LLP. Ministry of Health, Review of Island Health's IHealth Electronic Health Record System; 2017. Available from: www.health.gov.bc.ca/library/publications/year/2017/review-of-Island-Health-IHealth-electronic-health-record-system.pdf.

[90] McCartney M. When a crisis is the predictable outcome of poor policy making. British Medical Journal. 2018;360(k90).

[91] Cullen WD. The Public Inquiry into the Piper Alpha Disaster. Stationery Office Books; 1990.

[92] Cullen WD. Health and Safety Executive: Ladbroke Grove Rail Inquiry. The National Achives; 2001. Available from: discovery.nationalarchives.gov.uk/details/r/C15859.

[93] UK Government. Grenfell Tower Inquiry terms of reference published; 2017. Available from: www.gov.uk/government/publications/grenfell-tower-inquiry-terms-of-reference-published.

[94] UK Government. Independent Review of Building Regulations and Fire Safety: publication of terms of reference; 2017. Available from: www.gov.uk/government/news/independent-review-of-building-regulations-and-fire-safety-publication-of-terms-of-reference.

[95] Fu K. Archimedes Center for Medical Device Security; 2018. Available from: www.secure-medicine.org/publications.

[96] Koppel RJ, Gordon S, editors. First, Do Less Harm: Confronting the Inconvenient Problems of Patient Safety (The Culture and Politics of Health Care Work). ILR Press; 2012.

[97] Jiaquan X, Murphy SL, Kochanek KD, Bastian BA. Deaths: Final Data for 2013. National Vital Statistics Reports. 2013;64(2).

A Estimates of preventable error

James estimated preventable fatality in hospitals [1], that is, deaths that occur because of preventable error or mismanagement. The US Centers for Disease Control and Prevention (CDC) publishes US death statistics [97]. We adjust all hospital-related deaths (excluding accident and suicide fatality rates) equally, and collect all the preventable hospital deaths into one factor to obtain the chart shown in Figure 1.

Figure 1 also shows car accident fatality (driver, passenger and pedestrian combined) from US NHTSA figures for 2013. Car accident fatalities, which are taken seriously, are only a tenth of preventable hospital fatality.

The horizontal lines in the chart are two estimates from James's paper: the higher 16.9% (A) is based on a compensated estimate, which James believes closer to the true figure, and the lower 8.1% (B) is a mean from 4 studies. Note that the percentages shown have as denominator the 2013 US population, but the James paper uses absolute numerical data from historical surveys (2006 on). Since incidents are likely to have increased in proportion with population growth, the percentages may be too low by about the corresponding growth in population since the studies were undertaken; the underestimate may be around 6%.