The ironies of automation … still going strong at 30?

Gordon Baxter, John Rooksby, Yuanzhi Wang and Ali Khajeh-Hosseini

School of Computer Science

University of St Andrews
St Andrews
UK
{Gordon.Baxter, jrnr Derek.Wang, ak562}@st-andrews.ac.uk

ABSTRACT

Motivation – Bainbridge highlighted some of the ironies of automation 30 years ago and identified possible solutions. Society is now highly dependent on complex technological systems, so we assess our performance in addressing the ironies in these systems.

Research approach – A critical reflection on the original ironies of automation, followed by a review of three domains where technology plays a critical role using case studies to identify where ironies persist.

Findings/Design – The reliability and speed of technology have improved, but the ironies are still there. New ironies have developed too, in cloud computing where the cheaper cost of computing resources can lead to systems that are less dependable when developers bypass company procedures.

Research limitations/Implications – The work relies on published or reported cases. This makes it difficult to precisely determine how widespread the issues are.

Originality/Value – The research re-iterates the importance of the need to regularly consider the ironies of automation in systems development so that we can mitigate against any potential adverse consequences.

Take away message – The more we depend on technology and push it to its limits, the more we need highly-skilled, well-trained, well-practised people to make systems resilient, acting as the last line of defence against the failures that will inevitably occur.

Keywords

Resilience, human factors, ergonomics, systems engineering

1 INTRODUCTION

This year (2012) marks the 30th anniversary of Lisanne Bainbridge's presentation on the ironies of automation at the IFAC/IFIP/IFORS/IEA conference on analysis design and evaluation of man-machine systems (subsequently published as Bainbridge, 1983). Banbridge's paper continues to be regularly cited in discussions about the issue of keeping the human in the loop. In the paper Bainbridge essentially provided a summary of the issues that had been seen in the process industries and aviation. The research predated much of the work on distributed systems and widespread adoption of the personal computer as well as the advent of the Internet.

The way that people work, and the technology that they use to carry out their everyday work have both changed significantly since the paper was originally written. As we build systems (and systems of systems, Maier, 1998) that are increasing in both size and complexity, it therefore seems apposite to look back at the ironies of automation, to see whether they can still offer us any lessons for the future, and to consider whether recent technological developments may have some new ironies in store for us.

We begin by providing a recap of the ironies of automation that Bainbridge originally highlighted. We then consider the way that technology has developed over the past 30 years, and what effect this has had on the ironies of automation. We focus our discussions on three domains that are the subject of ongoing interest because of the central role occupied by technology: aviation; financial trading; and cloud computing. For each domain we illustrate our discussions with an analysis of a case study of a situation where things went wrong. We highlight some of the ironies in each case study, and discuss the ironies more broadly for each domain, highlighting where some new ironies are starting to emerge. We finish with a general discussion of the persistence of the ironies of automation and what we can do to mitigate against them.

2 THE IRONIES OF AUTOMATION (1982)

In her original paper, Bainbridge focused mainly on issues associated with monitoring and control activities in the process industries (chemical production, steel manufacturing and so on), with some examples from the flight deck in aviation. Some of the tasks that the operators were required to perform were automated more than others, such as the carrying out of routine day to day process operations, as well as process (and plant) shut-downs. One of the key factors in deciding whether a particular task could be automated was the predictability of the behaviour of the industrial process that was being controlled.

At that time operators were often perceived by system designers as a major source of variation and unpredictability in system performance. Many system designers therefore believed that the human operators should be (largely) removed from the system. This was in spite of the large body of evidence on socio-technical systems that had been amassed over the previous two decades (Emery & Trist, 1960; Eason, 1988) showing the importance of taking account of the interdependencies between people and technologies that are intrinsic to getting work done.

Bainbridge discussed the ironies under the separate headings of manual control skills, cognitive skills and monitoring. There are some overlaps because the underlying issues cut across the topics of skill learning and retention, situation assessment and vigilance.

2.1 Manual control skills

When automation is introduced, the operator is left with two basic types of task. The first is to monitor the automation, and when it fails to operate as expected, to intervene or call on someone more experienced to carry out the intervention. This requires manual control skills.

The first irony was that as tasks were automated, and hence taken away from the operators, this reduced their manual control skills. The operators were in a position where these skills were highly developed, and could effectively be carried out automatically, with little need for control, and hence required few attentional resources. In many situations the operators were practising open loop control. When tasks get automated, however, the operators get fewer chances to practise these skills. The net result is that these skills deteriorate, so the operators become slower at doing the tasks as they end up regressing towards closed loop behaviour in which they actively rely on feedback on their actions before deciding what actions to take next.

The second irony is that when things go wrong, it is invariably up to the operators to intervene manually. These situations are often unanticipated, so the operators have to perform non-routine actions in order to bring the process back under control, or to a safe state. The net effect is that the operators need to be more skilled in order to devise and carry out the required actions, and have to be less heavily loaded so that they can identify and perform the required actions in the first place.

2.2 Cognitive skills

The second basic type of task that is left to operators is the diagnosing of problems when they occur. The operator needs to decide whether the identified fault means that the process has to be shut down, or whether it could be safely recovered, for example. In order to carry out the diagnosis, the operator relies on the use of cognitive skills which Bainbridge divided into skills associated with long term knowledge, and what she called working storage (to distinguish it from short term memory).

2.2.1 Long term knowledge

Operators mostly develop their skills on the job, learning through practice which strategies to apply in particular situations. New strategies can be developed as they encounter new situations as long as they have a good understanding of the process they are controlling. In order to retrieve the appropriate knowledge efficiently it has to be used frequently, however, and this type of knowledge only develops with use and feedback based on applying it. The fear was that new operators would not have the opportunity to develop this knowledge, because they would become detached from the underlying process which they are trying to control.

2.2.2 Working storage

Operators make decisions are made based on the current context. This context is often encapsulated in a mental model (Moray, 1999), which the operator updates over time as the situation changes. The shift handover period, for example, provides one way in which operators can get up to speed with how the process is currently operating. If the operator has to intervene quickly on an automatically controlled plant, however, they will only be able to make decisions and take actions based on a minimal amount of information until they have had the chance to investigate further and consider the available options.

2.3 Monitoring

Bainbridge highlighted that monitoring, superficially at least, seems to be a straightforward task: the operator simply calls in their supervisor if things are not behaving as expected. This just transfers the ironies related to manual skills and cognitive skills to the supervisor, however.

Most of the time the process or system being controlled will run quite smoothly, and very little will happen, which raises vigilance issues. People find it difficult to maintain effective visual attention for more than about 30 minutes when the information they are looking at is largely unchanging. This makes it harder for operators to detect any abnormalities, so an automated audible alarm system is normally used. Which raises the question of who checks that the alarm system is working?

When automation is introduced it is usually because it is perceived that it can do a better job than the human. It is therefore ironic that the operator has to monitor the system to make sure that it is working correctly. This raises two problems.

The first is that the operator will require specialised knowledge—acquired through training, or dedicated displays—in order to be able to monitor the system effectively. This is particularly true where complex modes of operation are involved.

The second is that the system will normally be processing more information at a faster rate than humans can, in order to make decisions. It therefore becomes impossible for the operator to adequately track the system's behaviour in real time. Instead, the operators will only be able to check the system at a higher level of abstraction.

2.4 Solutions

All of the recommendations for solutions proposed by Bainbridge were flagged as being highly dependent on factors, such as the size, complexity and speed of the process, and the particular skills and abilities of the operator. Somewhat ironically, several of the solutions were technology based, such as multiple levels of alarm systems, and displays showing alarm boundaries. It was also deemed important that the technology fail in obvious ways, so that the operators can quickly spot it.

When considering the issue of whether to shut a system down in the event of a failure, Bainbridge suggested this would depend on the type of system: an aircraft should not be automatically shut down instantaneously, whereas it would be more desirable on a nuclear power plant. Where time allows for a system to be manually shut down, the operators need to be kept up to speed with how to do this through regular training and practice.

One way of keeping operators' manual skills up to date would be to allow them to manually operate the system on a regular basis (or at least in a simulator). Where decisions have to be made on the basis of changes that happen over time, any simulators used in training also need to be similarly dynamic. Any training can invariably only help in the development of general strategies—how can you train someone to deal with faults that are completely unknown?—expecting the operators to deal with unfamiliar events by following operating procedures is inadequate. This is another irony in that operators that are trained to follow procedures are then expected to provide intelligence in the system by adapting the procedures to the local contingencies, and to fill in any gaps in the procedures.

If a system is frequently generating alarms, the operators will become quite experienced with dealing with them in a routine manner. From this Bainbridge points out the final irony which is that the best automated systems which rarely require manual intervention are the ones that require the most investment in operator training to make sure that the operators can respond appropriately when things do go wrong.

3 FROM 1982 TO 2012

There are now very few areas of society where technology does not play a major role. We have reached the point where we are building systems, and systems of systems—where one system comprises many other systems that may have been designed and/or operated independently—of increasingly large scale.

The complexity of systems is also getting to the point where it is becoming more and more difficult to understand how the system really works. These large scale complex IT systems are almost invariably socio-technical systems that are operated by teams of people. The technological part of the systems has become increasingly reliable, offering up to 99.999% reliability. However, when the technology goes wrong—which it invariably will at some point—we continue to rely on people to step in and save the day, without giving them the resources to be able to do so. So we still find ourselves looking at situations where the ironies of automation prevail. In this section we discuss the ironies using examples taken from domains where technology plays a critical role: aviation, financial trading, and cloud computing.

3.1 Aviation

The aviation industry has been at the forefront of adopting automated technologies over the past 30 years. Part of the motivation for greater use of technology came from the need for increased levels of safety as the volume of air traffic continued rising (Orlady & Orlady, 1999). Some of the burden for handling safety and efficiency has been passed to the automation, such as the detection of other air traffic in the aircraft's vicinity, which is handled by the Traffic Collision Avoidance System (TCAS).

The net effect of the new technology on safety has not been clear-cut. There has, however, been a stable downwards trend in aviation accident numbers since 1992 despite the fact that the number of passengers has risen throughout the period (Ranter, 2007).

Many pilots openly acknowledge that modern, glass cockpit aircraft—where digital displays replaced the old analogue dials and meters—are easier to fly than the older aircraft. When you look deeper, however, you find that the pilots' workload has moved, rather than been reduced: pilots now have to carry out more supervisory tasks.

In the aircraft cockpit there is limited space for introducing new technology, so you will often find single pieces of equipment that have several operating modes. These modes may be controlled by a simple switch, but it may not always be immediately obvious from the displays which mode that equipment is currently operating in. The lack of visibility of concurrent operating modes and automated tasks has made it more difficult to predict the outcome of particular actions, with pilots reporting increased problems in understanding and anticipating aircraft behaviour, and in tasks such as programming the Flight Management System (Rudisill, 1995). The corollary of this is pilots are spend increasing amounts of time with their heads down, interacting with the automation, instead of concentrating their efforts on their primary task of flying the aircraft. Pilots therefore end up expending more time and effort learning how to manage the new technology. The traditional mantra of training pilots to aviate, navigate, communicate has consequently been extended to add *manage systems* to their core set of skills. The problems that pilots have to deal with can be illustrated by the crash landing that happened at Nagoya airport in 1994.

3.1.1 Nagoya Crash Landing

On April 26th 1994, an Airbus A300-600 glass cockpit aircraft left Taipei (Taiwan). Two hours and 22 minutes later, the aircraft crash-landed tail-first at Nagoya in Japan, killing 264 people (Ministry of Transport, 1996). The causes of the crash included a complex interaction between the aircraft's Go-around mode, autonomous inputs from the trimmable horizontal stabilizer (THS), and variations in the aircraft's thrust.

The First Officer (FO) was flying during the final stage of the approach to the airport. At 100s before impact at an altitude of 1070ft, the FO had erroneously engaged Go-around mode, which causes the aircraft to apply maximum thrust and climb rate in order to regain altitude. It is often used when an aircraft misses its landing. The Captain noticed the FO's mistake five seconds later, and called out for the mode to be disengaged. This was not done, and as a result the aircraft quickly climbed above the intended glide path. The FO tried to counteract the problem by lowering the aircraft's nose and decreasing the thrust. Although the aircraft resisted the nose-down command, it did level off temporarily at 1040ft, and the FO managed to throttle back the engines. At 87s prior to impact, the autopilot was engaged which caused the THS to take control of the aircraft's attitude. The aircraft started to pitch up again 68s from impact; 4s later, the FO disengaged the autopilot.

At 48s from impact, the angle of attack—the angle between the wing and the airflow—was still increasing, giving the aircraft greater lift. The Alpha-floor protection mechanism triggered an automatic recovery from the near-stall conditions resulting from the aircraft's low air speed, and the aircraft began climbing again at 570ft. The Captain took over the controls but was unable to lower the aircraft's nose to halt the climb, and subsequently expressed his puzzlement and worries about the aircraft's behaviour. The increasing nose-up attitude could not be controlled and the situation was exacerbated by the thrust being increased and decreased several times. At 19s before impact, the A300 went into a stall at a 52° nose-up attitude. The crew attempted several corrective actions on the ailerons and rudder unsuccessfully and the aircraft crashed tail-first.

The irony here was that the pilot did not realise that the aircraft was in Go-around mode and so ended up fighting the automation to try to regain control of the aircraft. It was also clear from the captain's verbal expressions of puzzlement about the aircraft's behaviour that they could not understand what it was doing, making it a classic example of an automation surprise (Sarter, et al., 1997).

The situation was further exacerbated by the flight deck technology in that automatic yoke-force disengagement of the autopilot was inhibited below 1500ft. Had it been enabled, the autopilot would have disengaged when the crew applied pressure on the yoke to bring the aircraft's nose down, which would have allowed the crew to bring the aircraft back under control.

3.1.2 The ironies in aviation

The aviation industry has generally been very good at dealing with automation issues. The report by the FAA's Human Factors team, *The interfaces between flightcrews and modern flightdeck systems* (1996), for example, captured many of the important issues associated with glass cockpits in the mid 1990s. Flight deck technology has progressed considerably since then, but the skills to deal with the advances in technology have not, and manual skills have been eroded too as the pilots rely increasingly on the technology to fly the aircraft. This has led to recent calls for changes to the recurrent training that pilots have to regularly undergo in order to reconcile pilot skills with the newer technologies (Learmount, 2011). There are several classic ironies here. Manual control skills, such as being able to recover from a stall, and carrying out a go-around in the event of a missed approach are being eroded.

Where new technology has been introduced, it does not always support the flight crew in the way they carry out their tasks (Baxter, et al., 2007). The irony here is that whilst the new technology helps the pilots to develop mental models of the situation, these models may be inconsistent with the state of the world because the automation does not present the pilots with the information they need in an obvious and timely manner.

3.2 Financial Trading

Since the mid-1980s automation has become increasingly widespread in the world's financial markets. Deregulation of the market in London in 1986 (often referred to as The Big Bang) led to a movement away from face-to-face trading on the market floor to on-line trading by telephone and computer from separate dealing rooms. In traditional markets, the success of a trader depended on their knowledge of the numbers, knowing when to continue and when to exit, and making timely decisions. It has always been difficult for an individual to monitor and anticipate the market, rather than just react to events as they happen. One of the driving forces behind automated algorithmic trading was that the technology could monitor larger amounts of information more quickly than people could, thereby facilitating faster decisions about buying and selling as prices and markets fluctuated. The human traders may now be based in offices anywhere in the world, but to minimise execution time the automatic trades are normally carried out on servers located close to the digital stock exchange.

The majority of financial trading is now automated. A particular style of trading has also emerged, known as high frequency trading (HFT). In HFT an automated system may hold a particular position for a matter of seconds. If it can make a net profit of even a few pennies in those few seconds, this can quickly lead to a steady stream of income. In the US markets it has been estimated that HFT could yield an annual income of at least of the order of $10bn (Kearns, et al., 2010), although this is quite small compared to the overall trading volume (about $50 trillion in 2008).

The human trader's role is now largely one of setting strategies and monitoring their execution. Even where the decision making is done by a human, the execution often still needs to be carried out algorithmically: there is a risk, however, in the time a trade takes to execute, because another algorithm may have spotted that the trade is taking place and intervene to make its own profit. The sorts of problems that have arisen in financial trading can be illustrated by the crash that happened on Black Monday and, more recently, the Flash Crash.

Black Monday and the Flash Crash

When the world's stock markets crashed on Black Monday (October 19th, 1987), many people laid the blame at the door of the technology. This was despite the fact that the trading software applications were effectively making the same decisions that human traders would have done in the same situation. At that time it was common practice to blame program trading whenever there were rapid price movements (particularly downwards). A more detailed analysis of events, however, showed that the biggest falls in prices did not occur at the heaviest times of program trading, so program trading was not solely to blame for the crash (Furbush, 1993). As with most accidents, the real explanation is more complicated and remains a topic of debate, but certainly involved program trading, overvaluation, illiquidity, and market psychology.

More recently, the so-called Flash Crash happened on the May 6th, 2010. The US's Dow Jones Industrial Average, which was already down on the day by over 300 points, fell more than 600 points in the five minute period starting at 2:42 pm. This effectively wiped $1 trillion from the value of the market. The Dow subsequently recovered most of the 600 points in the ensuing 20 minute period. Whilst this was the largest within day fall on the Dow, it was the speed at which it happened that really stunned most people.

Several reasons for the Flash Crash have been put forward. Some of these have been debunked, such as the idea that somebody inadvertently sold more stock in Proctor & Gamble than they had meant to. Others are disputed, such as the fact that the sale of 75,000 E-mini S&P contracts caused a major dislocation in the futures market. It took the US Securities and Exchange Commission (SEC) and Commodity Futures Trading Commission (CFTC) nearly five months to publish its official report into the Flash Crash (CFTC/SEC, 2010), and when it appeared it was widely criticised for its explanation of events.

Irrespective of the underlying causes, prices only stabilised when the Chicago Mercantile Exchange's Stop Logic Functionality was triggered to prevent a cascade of further falls in the price of E-mini S&P contracts. The five second pause in trading that it created was accompanied by a reduction in market pressures. Shortly afterwards the price of the E-mini contracts started to recover, and the Dow began to bounce back.

In the aftermath of the Flash Crash trading curbs, known as circuit breakers, were introduced in the US. These are intended to halt trading in any S&P 500 stock that rises or falls by more than 10% in a five minute period. On the day of the Flash Crash trades were only halted when they were more than 60% away from the reference price, and the process for breaking a trade was not clear to traders who were participating in the market.

The new circuit breakers halt trade for a five minute period. Initially the introduction of the programme of circuit breakers was limited to the S&P 500 stocks listed on the New York Stock Exchange. The idea has subsequently been expanded to other areas of the market (using trigger levels appropriate to the particular market).

Whilst the circuit breakers may prevent the exact same event from happening again, they do not remove the risk of other sorts of crashes. One worst-case scenario is a Splash Crash, where a stock market event splashes out into the currency markets and beyond.

3.2.1 The ironies in financial trading

The lower limit for trade execution times is currently around 10μs (Haldane, 2011). The knock-on effect of the reduction in these times is that the role of the human trader has changed from one of executing trades to one of configuring algorithms, monitoring trades and evaluating results. The classic irony here is that the human cannot monitor the trades in anywhere close to real time, so they have to monitor them at a higher level of abstraction, but even then still need time and resources available to process the information.

The other classic irony occurs when things fail. The speed of the trades makes it impossible to immediately detect a single failure until it has become large enough to be noticeable by a human. In the time it takes to diagnose and repair the failure, however, many more trades may have been executed, and possibly have exploited that failure. Haldane (2011) suggests the possibility of imposing minimum resting periods on all trades, which would place a lower level time limit on each trade. He argues that this would help restore the balance between market efficiency and market stability; to date regulatory changes have tended to favour market efficiency.

3.3 Cloud computing

The advent of cloud computing has created another area where automation is having an increasingly significant impact on people, organisations and work. The US National Institute for Standards and Technology defines cloud computing as '… a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.'

(National Institute of Standards and Technology, 2011). It goes on to briefly summarise the essential characteristics of cloud computing; the different types of service models—infrastructure, platform, and software as a service—and the deployment models—private, community, public and hybrid clouds.

Cloud computing can lead to radical changes in IT infrastructure within an organisation. Although it will never completely replace all of an organisation's IT, it can drastically reduce it to the point where all that the organisation needs is a device for each user to run some client software (usually within a web browser, or an app), and some way of connecting each of the devices to the Internet. All of the other parts of the infrastructure (storage, processing power, and software application packages and services) reside in the cloud, and are operated by one of the many Cloud Provider companies. Through economies of scale, the Cloud Provider companies can make computing resources widely accessible, on demand, at very low cost. The sorts of problems that can arise when the technology fails can be illustrated by the outage that happened to Amazon Web Services (AWS) in 2011.

3.3.1 The Amazon Web Services outage

AWS is one of the major cloud providers and has several data centres located around the world. In April 2011 they suffered a major outage mostly affecting customers of Amazon's Elastic Compute Cloud (EC2) services on the East Coast of the US.

AWS achieves high dependability through the use of mechanisms such as regions and availability zones, each of which runs on its own distinct independent infrastructure and provides geographic and physical isolation from failures. Each of their data centres is also configured using high levels of redundancy to achieve dependability. Within the availability zones each Elastic Block Store (EBS) node, which offers persistent storage, is protected by a peer-to-peer based, fast failover strategy. Each storage volume is replicated within the same availability zone. If one of the volumes ever gets out of step or becomes unavailable, new replicas are provisioned immediately.

EBS operates over two networks: a primary, high bandwidth network; and a secondary, lower bandwidth network which is used for node replication. If the primary network gets swamped with traffic, reliability is maintained by communicating over the secondary network. If a node loses connectivity with its replica, it assumes that that node is down.

At midnight on the April 21st, 2011, during a routine upgrade operation, an engineer configured a router in such a way that all network traffic was switched to the low capacity, secondary network. When nodes started losing contact with their replica, they assumed the mirror node had failed, and so they started trying to create a replacement.

AWS engineers tried to mitigate the problem by disabling node creation, which appeared to work for a while. The nodes do not have a fast back off rate so they continued trying to locate a replica. Problems were made worse when a very low probability bug associated with a race condition in the availability zone began occurring frequently. The control panel had a long time out for retry requests, which led to thread starvation which, in turn, led to the problem cascading into other availability zones. AWS ended up having to basically isolate the whole data centre to contain the problem, and took it offline 12 hours after the problem occurred.

The recovery process lasted about 24 hours. AWS physically relocated capacity from another data centre to the affected one. It took a long time to integrate the resources and then re-replicate the nodes.

The outage, which lasted 36 hours, led to a lot of unhappy customers. Some customers lost their servers (the web sites for Foursquare, Reddit and Quora all became unavailable, for example); others struggled to understand the implications for their business; and many customers particularly criticised the failure of AWS to communicate what was going on.

The classic irony with the AWS outage was that most of the customers who suffered data losses had either lost, or never had expertise in system administration and understanding the design of AWS. Data loss is guaranteed by AWS as long as you use the AWS cloud correctly. In other words, you have to know how it all works, and design your architecture so that it is robust to the kinds of outages described above through redundancy (e.g., by using multiple virtual machines).

3.3.2 The ironies in cloud computing

Cloud computing is different to the domains that Bainbridge originally used to highlight the ironies of automation. In cloud computing the applications are much more likely to be business or mission critical than safety critical. Despite this, it is easy to identify several of the classic ironies, and even some new ones.

Although moving to the cloud may create the impression that you are passing responsibility for your infrastructure to the cloud provider, it turns out that things are not quite that simple. In the traditional model of computing you may have bought hardware (and support) and separately bought software (and support). In cloud computing you can buy packaged solutions from software providers who will provide you with computing resources to run the software which they have purchased from a cloud provider. This makes it harder, if not impossible, to monitor the system for potential problems. It can be very challenging to maintain end-to-end system dependability when the management of responsibilities for the overall system become distributed among several stakeholders. When you spot a problem as an end user, for example, the only thing you may be able to do is report it to your software provider, who will then pass it on to the cloud provider to deal with.

There is one major new irony in cloud computing too, which relates to the low cost of using cloud computing resources. The fact that the resources are widely available at very low cost means that they can be purchased by anyone armed with a credit card. This allows anyone to develop and deploy software applications that are potentially unsafe, insecure, and undocumented by bypassing a company's standards and processes that are supposed to assure the quality of software that they produce. At one stage there were reports of one company in the oil and gas industry where a large number of system developers used personal AWS accounts to develop, test and host a new customer facing website (Fellows, personal communication). In a similar situation a person in another organisation set up a personal AWS account to develop some software (Dierickx, personal communication). The developer followed AWS' best practice, but did not document the development process. The information needed for authentication on AWS was stored in a file on the developer's desktop. The software was completed and launched into production without testing. The real problems only surfaced when the system failed after the developer left the company. The developer's old employer could not get access to AWS to fix the problem because the AWS account was tied to a personal credit card. This meant they had no visibility or control over the account that was using the cloud services. The situation was further exacerbated by the lack of documentation, and the fact that all of the critical information needed to access the services was stored on the developer's laptop.

4 PLUS ÇA CHANGE, PLUS C'EST LA MEME CHOSE…

So here we are 30 years after the ironies of automation were originally highlighted. In the interim period a significant body of research has been developed in human factors and ergonomics that is relevant and directly applicable to systems development. In some cases the results come from empirical studies, and in others it has been through the analysis of both incidents and accidents.

The notions of human centred design and user centred design are not new, but they are still not as widely practised as they should be. In general the uptake of user-centred methods has been rather disappointing. Eason (2001), for example, found that none of the 10 most widely advocated user-centred methods were being commonly used.

Part of the underlying problem lies in the fact that system developers are still not taking appropriate account of the people that will ultimately use their systems. We are not advocating that software developers become human factors experts. Instead we fully support the notion that systems development is an integrated interdisciplinary endeavour, where disciplines including software engineering, hardware engineering, human factors engineering and ergonomics, psychology and sociology all potentially have something useful to contribute. Currently the technical engineers (hardware and software) are still dominating and driving much of systems development.

The recent report from the US Committee on Human-System Design Support for Changing Technology (Pew & Mavor, 2007) has pointed the way towards better human-system integration. The report provides a model of the system development process that takes explicit account of human factors at all levels from human-machine interaction up to the constraints and limitations imposed by regulatory and legal authorities.

We are still seeing too many examples of the ironies of automation. There are two fundamental ironies that are likely to persist for some time to come. The first is that as systems become more and more dependable (like cloud computing, for example) the opportunity for users to manually work with the technology to learn and apply operating skills will continue to be reduced. When the technology fails, as it inevitably will, because no technology is 100% reliable, the operators will therefore not have the skills to diagnose and solve the problem in a timely manner.

The second irony is that as we continue to push the limits of technology, particularly in terms of speed—Haldane (2011) calls this "[t]he race to zero" in financial trading—it also becomes harder for the operators to monitor what the technology is doing in real time. Again, the net effect is that when the technology fails to work as expected, the operators will not be able to immediately intervene to diagnose and rectify the situation.

Most systems nowadays are socio-technical systems that involve people working together, and with technology. The resilience of these systems is heavily dependent on the people who use them to act as the last line of defence when the technology inevitably fails, possibly in ways that were not expected by the designers. People are good at stepping in to save the day because they are inherently flexible and adaptable. In order to do so, however, they have to:

- be taught the skills to do this;
- be allowed to practice (and update) these skills on a regular basis; and
- be provided with the appropriate information by the technology in a timely manner.

Only in this way can we hope to finally lay the ironies of automation to rest.

5 ACKNOWLEDGMENTS

This work was supported by funding from the UK's EPSRC for the Large Scale Complex IT Systems (LSCITS) project, and from the Scottish Funding Council for the Creating High Value Cloud Services project.

6 REFERENCES

[1.] Bainbridge, L. (1983). Ironies of automation. *Automatica, 19*, 775-780.

[2.] Baxter, G., Besnard, D., & Riley, D. (2007). Cognitive mismatches in the cockpit: Will they ever be a thing of the past? *Applied Ergonomics, 38*(4), 417-423.

[3.] CFTC/SEC (2010). *Findings Regarding the Market Events of May 6, 2010: Report of the staffs of the CFTC and SEC to the joint advisory committee on emerging regulatory issues*. Washington, DC: CFTC/SEC.

[4.] Eason, K. (1988). *Information technology and organisational change*. London, UK: Taylor & Francis.

[5.] Eason, K. (2001). Changing perspectives on the organizational consequences for information technology. *Behaviour & information technology, 20*(5), 323-328.

[6.] Emery, F. E., & Trist, E. L. (1960). Socio-technical systems. In C. W. Churchman, & Verhulst, M. (Ed.), *Management Science Models and Techniques* (Vol. 2, pp. 83-97). Oxford, UK: Pergamon.

[7.] FAA Human Factors Team (1996). *The interfaces between flightcrews and modern flightdeck systems*. Washington, DC: Federal Aviation Authority.

[8.] Furbush, D. (1993). Program Trading *The Concise Encyclopedia of Economics*. Indianapolis, IN: Library of Economics and Liberty.

[9.] Haldane, A. G. (2011). The race to zero. Speech given at the International Economic Association Sixteenth World Congress, Beijing, China, 8 July 2011. Retrieved from http://www.bankofengland.co.uk/publications/Documents/speeches/2011/speech509.pdf on April 9th, 2012.

[10.] Kearns, M., Kulesza, A., & Nevmyvaka, Y. (2010). Empirical limiations on high frequency trading profitability. *The Journal of Trading, 5*(4), 50-62.

[11.] Learmount, D. (2011). Industry sounds warning on airline pilot skills. *Flight Global*. Retrieved from http://www.flightglobal.com/news/articles/industry-sounds-warnings-on-airline-pilot-skills-352727/ on April 9th, 2012.

[12.] Maier, M. W. (1998). Architecting principles for systems of systems. *Systems Engineering, 1*(4), 267-284.

[13.] Ministry of Transport (1996). *Aircraft Accident Investigation Commission. China Airlines Airbus Industries A300B4-622R, B1816, Nagoya Airport, April 26, 1994.* (No. Report No. 96-5): Ministry of Transport, Japan.

[14.] Moray, N. (1999). Mental models in theory and practice. In D. Gopher & A. Koriat (Eds.), *Attention and performance XVII: Cognitive regulation of performance: Interaction of theory and application* (pp. 223-258). Cambridge, MA: MIT Press.

[15.] National Institute of Standards and Technology (2011). *The NIST Definition of Cloud Computing* (No. NIST Special Publication 800-145). Gaithersburg, MD.

[16.] Orlady, H. W., & Orlady, L. W. (1999). *Human factors in multi-crew flight operations*. Aldershot, UK: Ashgate.

[17.] Pew, R., & Mavor, A. (Eds.). (2007). *Human-system integration in the system development process: A new look*. Washington, DC: The National Academies Press.

[18.] Ranter, H. (2007). Airliner accident statistics 2006. Retrieved from http://aviation-safety.net/pubs/asn/ASN_Airliner_Accident_Statistics_2006.pdf on April 9th, 2012.

[19.] Rudisill, M. (1995). Line pilots' attitudes about and experience with flight deck automation: Resutls of an international survey and proposed guidelines. In R. S. Jensen & L. A. Rakovan (Eds.), *Proceedings of the 9th International Symposium on Aviation Psychology* (pp. 288-293). Columbus, OH: The Ohio State University.

[20.] Sarter, N. B., Woods, D. D., & Billings, C. E. (1997). Automation Surprises. In G. Salvendy (Ed.), *Handbook of Human Factors and Ergonomics* (2nd ed., pp. 1926-1943). New York, NY: Wiley.