High-Pressure Steam Engines and Computer Software*

Nancy G. Leveson
Computer Science & Eng. Dept., FR-35
University of Washington
Seattle, WA 98195

Even though a scientific explanation may appear to be a model of rational order, we should not infer from that order that the genesis of the explanation was itself orderly. Science is only orderly after the fact; in process, and especially at the advancing edge of some field, it is chaotic and fiercely controversial.

— William Ruckelshaus [33, p.108]

The introduction of computers into the control of potentially dangerous devices has led to a growing awareness of the possible contribution of software to serious accidents. The number of computer-related accidents so far has been small due to the restraint that has been shown in introducing them into safety-critical control loops. However, as the economic and technological benefits of using computers become more widely accepted, their use is increasing dramatically. We need to ensure that computers are introduced into safety-critical systems in the most responsible way possible and at a speed that does not expose people to undue risk.

Risk induced by technological innovation existed long before computers; this is not the first time that humans have come up with an extremely useful new technology that is potentially dangerous. We can learn from the past before we repeat the same mistakes. In particular, parallels exist between the early development of high-pressure steam engines and software engineering that we can apply to the use of computers in complex systems.

The Problems of Exploding Boilers

Great inventions are never, and great discoveries are seldom, the work of any one mind. Every great invention is really either an aggregation of minor inventions, or the final step of a progression. It is not a creation but a growth *— as truly so as that of the trees in the forest. Hence, the same invention is frequently brought out in several countries, and by several individuals, simultaneously. Frequently an important invention is made before the world is ready to receive it, and the unhappy inventor is taught, by his failure, that it is as unfortunate to be in advance of his age as to be behind it. Inventions only become successful when they are not only needed, but when mankind is so advanced in intelligence as to appreciate and to express the necessity for them, and to at once make use of them.*

Robert H. Thurston
A History of the Growth of the Steam Engine (1883)

Hero of Alexandria, who lived around 60 AD, conducted some of the first known investigations into the use of steam for power. But it was not until the 16th and 17th centuries that the problem of pumping water out of mines changed the search for steam power from a diversity to a necessity. Many inventors attempted to harness this source of power, but Savery is usually credited as the first to produce and sell a workable steam apparatus. Then Newcomen designed a practical cylinder and piston engine around 1700 which is the forerunner of all subsequent steam engines.

In 1786, James Watt was working as an instrument maker at Glasgow University and was asked to repair a model of a Newcomen engine that was being used in a Natural Philosophy class. By one of those serendipitous coincidences of history, Watt had become friendly with several professors, including Dr. Joseph Black, a chemistry professor who discussed with Watt his recent discovery of the phenomenon of latent heat. Watt was unique among the early steam engine inventors in

*This paper was presented as a keynote talk at the International Conference on Software Engineering, Melbourne, Australia, May 1992 and is included in the proceedings.

having had direct and indirect contact with scientists who studied heat [17].

Watt decided he could improve on the Newcomen engine and patented several important ideas, including the separate condenser and the design of an engine producing rotating motion, at the same time as the industrial revolution was generating a demand for power on an unprecedented scale. With a successful manufacturer named Matthew Boulton, Watt came up with a design for a steam engine that was the leading edge of technological change in the last two decades of the eighteenth century. The application of steam power transformed industry in terms of output and productivity and produced even more revolutionary changes in transportation when it was applied to locomotives and ships.

The Boulton and Watt machines used low-pressure steam (from 5 to 15 psi), which limited both their efficiency and economy. Higher pressure (i.e., above atmospheric pressure) would have permitted more powerful and economical engines, but Watt opposed it on the grounds that it increased the danger of explosion and thus constituted an unacceptable risk.

Although Watt and Boulton resisted making high-pressure steam engines, their patent expired in 1800, and such engines soon made their appearance. Oliver Evans in the U.S. and Richard Trevithick in England almost simultaneously designed engines that dispensed with condensers and used steam directly to push a piston. These so-called high-pressure engines required greater than atmospheric pressure to work.

The first wide-spread application of the high-pressure engine, on steamboats, resulted in frequent and disastrous explosions: passengers and crew were blown up, scalded to death, hit by flying fragments of iron, and blown off steamers to drown. Accidents were also common in industrial uses of high-pressure steam. The early steam engines used inferior materials; they had low standards of workmanship; the mechanics lacked proper training and skills; and there were serious problems with quality control [10].

In the U.S, there were calls for professionalization and standardization of the training of steam engineers who typically had an informal and haphazard education. There was even a suggestion that the federal government establish an academy of steam technology. All of this came to naught and engineers continued for many years to be trained "willy-nilly" [30].

Watt's predictions about the danger of the new engine were correct. Cameron and Millard write:

> As the technology of steam power advanced, Watt found himself in an increasingly difficult dilemma: the trend toward greater efficiency and power also increased the risk of explosion. The technology that he had created escaped his control and became increasingly dangerous to life and property. Watt expected more accidents and deaths would result from adoption of high-pressure steam. The threat to public safety now overshadowed the public utility of steam power. . .
>
> But what could Boulton and Watt do? They were in no position to stem the economic forces that demanded more and more power from the steam engine. If they refused to develop the technology, many other engineers — most of them untrained and poorly skilled — were willing to take the risk of high-pressure steam. What they could do was to alert the public to dangers in the new technology and remind their fellow engineers of their special obligations to ensure public safety. Watt initiated the debate about the risks of the new technology and used his influence to press for safer, and better engineered, alternatives [10, pp. 6–7]

Watt's campaign against high-pressure steam along with some well publicized accidents slowed its adoption in England. Trevithick complained that his competitors had greatly exaggerated the risk and the accidents, writing:

> I believe that Mr. B. & Mr. Watt is abt to do mee every engurey in their power for the have don their outemost to repoart the exploseion both in the newspapers and private letters very differnt to what it really is [17].

A German supporter of high-pressure steam wrote in 1842 that the intense discussion of its defects and safety risks had clouded the issue of its advantages and had "disgusted the industrial community" [10].

The public pressure did force the makers of high-pressure steam engines to incorporate safety features [12]. The risk from this type of machine came from the boiler and not from the engine itself: It was the boiler that was exploding and causing most of the casualities. The technological development of boilers lagged behind the rapid improvement of the engines. Engineers quickly amassed scientific information about thermodynamics, the action of steam in the cylinder, the strength of materials in the engine, and many other

aspects of steam engine operation. But there was little scientific understanding about the buildup of steam pressure in the boiler, the effect of corrosion and decay, and the causes of boiler explosions [17]. High-pressure steam had made the current boiler design obsolete by producing excessive strain on the boilers and exposing weaknesses in the materials and construction of the boilers.

To counter this, engineers introduced two types of safety features: safety valves to reduce steam pressure when it reached a dangerous level and fusible lead plugs that were supposed to melt when the temperature in the boiler grew too hot because of the overheating of the steam. But these much publicized technological fixes did not solve the problems, and the number of explosions continued to increase. The fixes were unsuccessful because engineers did not fully understand what went on in steam boilers: It was not until well after the mid-century that the dynamics of steam generation was understood.

A second reason for the number of accidents was that engineers had badly miscalculated the working environment of steam engines and the quality of the operators and maintainers. Most designs for engines and safety features were based on the assumption that owners and operators would behave rationally, conscientiously, and capably. But operators and maintainers were poorly trained, and economic incentives existed to override the safety devices in order to get more work done. Owners and operators had little understanding of the workings of the engine and the limits of its operation.

While operators certainly did contribute to the problems, they were not solely responsible for them. Nevertheless, owners or operators received most of the blame for explosions; criticism was rarely leveled at the engineer who had designed the engine. As noted above, many of the engineers who took the risk of developing high-pressure steam technology were untrained and poorly skilled. Limited knowledge of the scientific foundations of their craft existed at that time. The personal standards of the inventor-engineer were the chief element in the safe operation of the engine, and Watt believed that engineers had a personal responsibility to ensure a safe and efficient steam engine and that they bore culpability in case of accidents.

Early opponents of high-pressure steam proposed regulations to limit its dangers by limiting the uses of the new technology. This idea met with little success. In the first half of the nineteenth century, governments were not disposed to interfere with private enterprise. The steam engine embodied the idea of success and was credited with "national progress almost unchecked, and of prosperity and happiness increased beyond all precedent" [10]. Many engineers argued that the social and economic gains of steam power were an acceptable trade-off for the risk involved. Typical was the response of U.S. Senator Thomas Hart Benton who, upon helping to defeat legislation to reduce boiler explosions on steamboats, remarked that masters and owners of steamboats were, with few exceptions, men of the highest integrity and that *he* had never met with any accident on a steamboat despite the fact that he traveled widely; upon boarding he was always careful to inquire whether the machinery was in good order [9].

But the dramatic increase in accidents that followed wide-scale introduction of steam engines was hard to ignore. An explosion of a steam-powered boat in England, followed by a series of industrial explosions, led to the creation of a Select Committee in 1817 to report on the dangers of high-pressure steam. The Committee began its report by acknowledging the great contributions of steam power to national prosperity and the drawbacks to interfering with private business. However, it noted that when public safety was endangered by "ignorance, avarice, or inattention . . . it becomes the duty of Parliament to interpose" [9]. The Committee recommended frequent boiler inspections, but their recommendations were not put into effect. Around the same time, the city council of Philadelphia was the first legislative body in the U.S. to take notice of the accidents and attempt to investigate. A report from the city council was referred to the state legislature where it died.

Accidents continued at an alarming rate during the 1830s and 1840s, which prompted more government attempts to limit risk. In the U.S., the Commissioner of Patents estimated that in the period of 1816–1848, a total of 233 steamboat explosions had occurred in which 2,562 persons had been killed and 2,097 injured, with property losses in excess of $3,000,000. The Franklin Institute, which had been founded in Philadelphia in 1824 for the study and promotion of the "mechanical arts and applied science," began a six-year study of boiler explosions. The first research grant of a technological nature by the U.S. government went to the Institute to defray the cost of the apparatus required for experiments in this study. In this instance, an invention and the accidents associated with it were pushing science. The result was a series of reports that exposed errors and myths in popular theories on the nature of steam and the causes of explosions, guidelines for the design and construction

of boilers to increase safety, and a recommendation that Congress enact regulatory legislation including requirements that engineers meet certain standards of experience, knowledge, and character [9].

As result of steamboat explosions, the prevailing bias against government regulation began to change. Laws were passed in both England and the United States requiring compensation for families of passengers killed in accidents due to neglect or default. There were, however, no inspection criteria included nor were qualifications set for engineers. The prevailing belief was that putting qualifications for engineers into effect was too difficult and that enlightened self-interest of entrepreneurs would guarantee the public safety. These laws failed to reduce the number of explosions.

Hundreds of newspaper editorials on the subject expressed the increased frustration of the public. The social costs of high-pressure steam engines versus the economic benefits were even treated in literature. Dickens wrote about them in Household Words [11], and, in the novel Gryll Grange by Thomas Love Peacock, a character remarks that "High pressure steam would not scatter death and destruction around them if the dishonesty of avarice did not tempt their employment, where the more costly low pressure engine would ensure absolute safety."

Public pressure plus a series of marine disasters killing hundreds more people finally forced the U.S. Congress to pass a law in 1852 that corrected the problems with steamboat boilers and reduced the number of steamboat accidents. This law was the first successful example of regulatory legislation in the United States, and it created the first U.S. agency to regulate private enterprise [9]. Unfortunately, similar legislation was not passed for locomotive and stationary boilers, and accidents involving the use of boilers in other than steamboats continued.

Watt and others were correct in their belief that new standards of precision and safety were essential in the design, manufacture, and operation of the engines. These high standards were finally enforced in Britain in the latter part of the nineteenth century, and boiler explosions in Britain fell dramatically. By 1905 there were only 14 deaths from boiler explosions in Britain as compared to 383 in the United States. Eventually, a majority of Americans also realized the necessity to enforce standards: Associations for the prevention of steam boiler explosions were formed; insurance companies were organized to insure steam equipment that was manufactured and operated with the utmost regard for safety; and, through the efforts of the American Society of Mechanical Engineers, uniform boiler codes were adopted [9].

Exploding Software?

We are now in the computer age and again are faced with a new technology for which there are great economic incentives to push the state of the art and to use this technology to control dangerous systems. Computers, like steam engines and electrical systems, give us the ability to accomplish things we could not accomplish before. And again, it appears that the risks could increase over time as computers take over more and more functions. One difference is the potential consequences of accidents: We are building systems and using computers to control them that have the potential for large-scale destruction of life and the environment. Even a few accidents may be disastrous in these systems.

It is therefore crucial that we use computers responsibly. Examining more closely the parallels from the past may provide some clues as to how to do this.

- *Boiler technology lagged behind improvement in steam engines themselves.*

Although computer hardware technology has advanced at an astounding rate, the development of software engineering has been slower. It has also been slower than required for the complex systems we want to build, like a space station or automatically-controlled nuclear power plants. There appear to be two ways to cope with this current shortfall.

The first is to fall back on a time-tested engineering principle: keep things simple and increase the complexity of what we are attempting to do slowly as we learn from our experiences. For example, Ontario Hydro recently became the first utility in Canada to obtain a license for a completely computerized nuclear power plant shutdown system. The software contains about 6000 lines of code and uses only the simplest, most straightforward coding techniques. Hardware fail-safe devices like watchdog timers and software self-checks are included to deal with some types of software errors. The software includes well-established safety design principles that were standard in the previous hardware shutdown systems. And because the software design is so simple, they were able to apply formal and informal verification and safety techniques [2, 4] in addition to using standard testing techniques to develop confidence in the software.

In contrast, the first computerized shutdown system in England, under licensing evaluation for the

Sizewell B reactor, has 100,000 lines of code, involves 300-400 microprocessors, and contains both control and shutdown functions [35]. This system not only goes beyond our ability to apply sophisticated software verification techniques, but it also violates the basic nuclear reactor safety design principle that requires complete independence of control and safety devices [1]. Safety design criteria of this type have been developed and proven over time — computer scientists need to be aware of them and engineers should think carefully before abandoning them: The design criteria represent knowledge accumulated by successes and failures in engineering over hundreds of years.

A second way to cope with the gap between software and hardware technology development also requires us to dampen somewhat our enthusiasm and confidence in computers. Although mistrust of computers has led to the use of hardware backup and fail-safe devices in the most critical systems, this mistrust is fading. Increasingly, existing hardware safety mechanisms and interlocks are being eliminated and computers substituted for monitoring and control. Engineers are deciding that the hardware safety interlocks and backups are not worth the expense, or in the case of aircraft, the extra weight, or they put more faith in software than in hardware reliability. This again violates a standard safety design principle that requires eliminating single-point failure modes, that is, the system should be built so that a single event (like a software error) cannot cause an accident. The Therac-25 is an apt example. The designers of this radiation therapy machine eliminated the usual hardware safety interlocks that are standard for linear accelerators of this type when they introduced computer control, believing that the hardware devices were no longer necessary. Instead, the interlocks and safety checks were implemented in software. After seven accidents between 1985 and 1987 involving massive radiation overdoses and four deaths, the company finally relented and put hardware safety devices on the machine [24].

We can be cautious in our use of computers to control dangerous systems without unduly hampering technological progress. James Watt campaigned against the use of high-pressure steam engines, yet he was only successful in delaying somewhat their use in Britain. In the 1880s, at the same time as the industrial world was struggling to cope with the rapid introduction of steam technology, similar issues arose with the introduction of high-voltage electricity. Another inventor, Thomas Edison, criticized the use of high voltage because of its complexity, poor reliability, and threat to public safety and began a campaign to alert the public of the dangers and of his belief that the size and impact of the risk would increase over time. Edison argued for a safe low-voltage electrical system that could quickly achieve public acceptance. Like Watt, he was only partially successful.

Another inventor-engineer, Elihu Thomson, also opposed high-voltage current as too dangerous. But instead of condemning the system and campaigning for its elimination, Thomson attempted to find a technological fix. He believed that several safety devices would greatly reduce the risk of accidents and lobbied for the need to engineer safe high-voltage systems. Thomson's argument was that a program of safety engineering would have commercial advantages in a highly competitive market for those companies with a technological lead in the construction of the safety devices.

Watt and Edison attempted to limit risk by arguing against the introduction of technology with tremendous potential benefits. In contrast, Elihu Thomson argued that we can limit risk by using simple, safe designs rather than limiting the uses of our technology or drastically inhibiting technological development. The Thomson approach is the more practical and more likely to be successfully applied to the use of computers in safety-critical systems.

- *There was little scientific understanding of the causes of boiler explosions.*

Like boilers, the scientific foundations of our field are still being developed. Changing from an art to a science requires accumulating and classifying knowledge. Although this is happening, more effort is being expended on new inventions and building tools for unproven techniques without rigorous scientific foundations. We need to carefully validate and assess our hypotheses using scientific principles.

Trial and error is a time-tested way of accumulating engineering knowledge. Engineers analyze the causes of failures and accidents and then take corrective measures to prevent or minimize their reoccurrence. The corrections eventually find their way into specifications, standards, codes, regulatory requirements, and what is considered to be good engineering practice. But this is a very slow way to accumulate knowledge. Early in the trial and error process, engineers start to look for analytical approaches. The brisk pace of technological development today is possible because of the foundational knowledge that has been developed about such things as mechanics, materials, and structures so that engineers do not have to evaluate

their designs only by building something and seeing whether it falls down over time.

There are two stages in the early years of a new technology: (1) exploration of the space of possible approaches and solutions to problems (i.e., invention) and (2) evaluation of what has been learned by this trial and error process to formulate hypotheses that can be scientifically and empirically tested in order to build the scientific foundations of the technology. Most of our emphasis so far has been in the first stage or invention; it is time now to give more attention to the second.

Invention is a worthy and necessary pursuit, but the most useful inventions are based upon or improved by scientific knowledge. Invention produces products, techniques, and tools. Science produces the knowledge and ability to evaluate and improve our products, techniques, and tools. Inventors use science to build better inventions, to know that they are better, and to compare them to what we already have. The gradual development of scientific knowledge led to the important patents by Watt that produced a practical steam engine. Further enhancement of basic knowledge about steam engines and boilers allowed the production of more effective and safer engines. Although rudimentary knowledge allowed the production and use of low-pressure steam engines, safe high-pressure engines required a deeper scientific foundation.

Software engineering inventions have provided leverage in building our current software systems. I do not want to denigrate what we have accomplished: We are building extremely complex systems, many of which work remarkably well a large amount of the time. But we may be straining at the limits of what we can do effectively without better inventions based on known scientific and engineering principles. And our early rapid progress may be slowing as we reach the limits of what we can accomplish on the basis of brute force. As an example, the late 1950s and early 1960's saw the development of very clever ways of building parsers for programming languages. But with the development of formal theories of grammars, parser generators became possible that eliminated the necessity of crafting a parser for each new compiler.

Similar needs exist in software engineering. Our greatest need now, in terms of future progress rather than short-term coping with current software engineering projects, is not for new languages or tools to implement our inventions but more in-depth understanding of whether our inventions are effective and why or why not. For example, we have a greater need to develop and validate the underlying principles and criteria for designing specification languages than to create more languages. We have a greater need to develop and validate basic design principles and to understand conflicts and tradeoffs between them than for more tools to specify designs. And we have a greater need to study the effects of different types of software development processes in real organizations and under different conditions than to create more languages for specifying processes.

Reseachers in some subfields of software engineering have been more conscientious in attempting to build their theoretical foundations. Testing is one such area, although they too have a long way to go. For example, testing researchers have defined theoretical ways of comparing testing strategies both in terms of cost and effectiveness (for example, [38]), formal criteria for evaluating testing strategies (for example, [16]), and axioms or properties that any adequacy criterion (rule to determine when testing can stop) should satisfy (for example, [37]). In general, theoretical foundations can provide (1) criteria for evaluation, (2) means of comparison, (3) theoretical limits and capabilities, (4) means of prediction, and (5) underlying rules, principles, and structure.

How will we build this foundation? It will require both building mathematical models and theories and performing carefully-designed experiments. In an abstract system, the elements are created by definitions and the relationships between them are created by assumptions (e.g., axioms and postulates). Many questions can be answered about abstract systems by using mathematics. In concrete systems (where some of the components are physical objects), establishment of the existence and properties of elements requires research with an empirical foundation since our knowledge of the physical laws involved are almost always incomplete.

The great power of the computer is that it is a general-purpose machine that can be changed into a special-purpose machine by the addition of a set of instructions (data) to accomplish that purpose. Software is an abstract design of a special-purpose machine that becomes a concrete design as soon as it is executed on a computer. Software then can and should be evaluated both as an abstract design and a concrete design. Furthermore, software is both a mathematical object and a human product. We cannot build effective tools or design techniques to help humans construct software without understanding the human problem-solving behavior involved in building software.

The empirical aspects of our field imply the neces-

sity for experimentation. As an example, formal methods have been proposed as a partial solution for the problems of ensuring safety, but there has been little validation of the hypotheses underlying these techniques. Does the use of formal methods result in fewer or different errors being made? Are the resulting programs more reliable? Are they safer? Are some techniques more effective than others? What type of training is necessary to use the techniques effectively? Is it more or less costly to use formal methods? Because the techniques must be employed by humans, it is not possible to answer these questions using only mathematical analysis; experiments involving humans will be necessary.

Intuition plays an important role in formulating hypotheses. But sometimes our intuition is misleading; we cannot stop with generating hypotheses (as we too often do now) no matter how much confidence our intuition allows us to place in them. Currently, we are applying techniques and even mandating them without validating that these work or that the underlying hypotheses and assumptions are valid (e.g., [3]).

When a physicist makes an erroneous claim, such as in cold fusion, the idea may stay around for a while on the fringes of the field. However, the insistence on repeatability and careful experimentation allows such claims to be dismissed by the scientific majority within a relatively short period of time. We need to insist on the same level of evaluation and proof with regard to claims about software engineering techniques and tools. Unfortunately, this is rarely done and our belief in silver bullets persist. Even after Brooks' and Parnas' carefully reasoned and widely-acclaimed papers [8, 27], we are still seeing claims that the silver bullet has been found.

I am not advocating that everyone stop the research they are doing in software engineering and start testing hypotheses and building foundations. Invention is a very important part of progress in engineering. Tools and techniques are needed for the serious problems we face today. But inventions that are based on established principles will be more effective in solving the complex problems we are attempting to solve. We need to recognize the unproven assumptions and hypotheses underlying our current software engineering techniques and tools and evaluate them in the context of what has actually been demonstrated about these hypotheses instead of what we would like to believe.

Like the exploding boilers, our ability to build safe software-controlled systems and to build effective software engineering tools to accomplish this will be enhanced by greater understanding of the scientific foundations of our craft.

- *The safety features designed for the boilers did not work as well as predicted because they were not based on scientific understanding of the causes of accidents.*

Not only do we not understand the underlying causes of software errors, but few researchers are examining the cognitive processes that underlie these errors. This has led to the development and use of methods to deal with errors that are based on erroneous underlying assumptions.

As just one example, claims of ultra-high software reliability in safety-critical systems and certification of these systems by government agencies have been based on the use of N-version programming (NVP). NVP involves separate teams writing multiple versions of the software. These versions are executed, and the majority answer (if there is one) is used. The technique is adopted directly from the hardware fault tolerance technique of N-modular redundancy where multiple copies of a component are connected to a voting circuit that selects the majority value.

The hardware technique was developed to cope with random failures, not with design errors. Despite this fact, NVP translates the approach into software terms and is used in most of the computerized commercial aircraft systems today as a way of supposedly achieving ultra-high software reliability. However, the few empirical studies performed on it did not test the underlying assumption of independence of failures and did not carefully analyze the data to determine whether ultra-high reliability was actually being achieved [23]. A series of experiments [6, 14, 22, 34] and a mathematical analysis [13] have cast doubt on these assumptions.

The latest approach by the proponents of this technique is to relabel it "software diversity" and to compare it to the established method of hardware design diversity although again the software technique does not satisfy the basic underlying assumptions. Diversity in hardware does not just happen; you have to design it in. Components with different failure modes, such as electronic and hydraulic components, are used in order to avoid common-mode failures. This crucial underlying assumption, that the components have different failure modes, is not satisfied by multiple software versions.

Not only do we need to validate that the assumptions underlying a software engineering technique satisfy the claims for it, but wishful labeling should be

avoided. Labeling a technique, e.g., "software diversity" or "expert system," with the property we hope to achieve by it (and need to prove about it) is misleading and unscientific. In the case of expert systems, a label like "production-rule system" (which, in fact, they were called before someone came up with the more sales-oriented label) would have been more scientific. Then those suggesting the use of this technique would more likely be required to prove that the system acts like an expert instead of this being taken as an axiom. In fact, psychological studies and theory have suggested that human experts do not make decisions in this way (e.g., [31, 28]): Much more sophisticated types of problem-solving are involved.

Related to proof by labeling is proof by definition, for example, defining fault tolerance as redundancy (another common practice) or defining safety as the use of protection (e.g., monitoring and shutdown) systems. In proof by definition, instead of embedding the property in the definition of a technique to achieve that property, the technique is embedded in the definition of the property. Two problems result. The first is the tendency to assume that the property has been achieved because the approach embedded in the definition is used, e.g., fault tolerance has been achieved because redundancy is used. The second is that the search for possible ways to achieve the property is limited to the embedded approach, e.g., if safety is defined as the use of protection systems to recover from hazardous states, other more reliable or effective techniques that eliminate hazardous states or minimize getting into them are not considered.

Unless we can develop a foundation of knowledge about human error in software development, it is doubtful that we will be able to design highly effective software development techniques to eliminate it or compensate for it. Moreover, we need to avoid equating humans with machines and ignoring the cognitive and human aspects of our field. Finally, we need to avoid proof by labelling or limiting solutions by our definitions and other such unscientific practices if we are to design, assess, and select the most effective safety and reliability enhancement techniques.

- *The introduction of safety devices for steam engines was inhibited not only by the lack of underlying scientific knowledge about boilers, but also by a narrow view of attempting to design a technological solution without looking at the social and organizational factors involved and the environment in which the device is used.*

A major airline, known for having the best aircraft maintenance program in the world, a few years ago introduced an expert system to aid their maintenance staff. The quality of maintenance fell. The staff began to depend on the computerized decision making and stopped taking responsibility and making their own decisions. When the software was changed to provide only information and only when requested, quality again rose. A similar example of this phenomenon has been found in aircraft: Hazardous situations have resulted when the introduction of computers increased pilot complacency and reliance and reduced situational awareness. The use of computers to enhance safety may actually achieve the opposite effect if the environment in which the computer will be used and the human factors are not carefully considered.

Some people have suggested that the solution is to remove humans from critical loops completely. However, in doing this, they are placing unjustified reliance on the ability of programmers to foresee all eventualities and correctly predetermine the best solution under all circumstances. And even highly automated systems need humans for supervision, maintenance, and operation.

Another aspect of technological narrowness is the emphasis on technical solutions over organizational and managerial considerations. Nearly every major accident of the past 20 years (for example, Three Mile Island, Chernobyl, Challenger, Bhopal, and Flixborough) involved serious organizational and managerial deficiencies. Management that does not place a high priority on safety can defeat the best efforts by the technical staff. In each of the recent accidents noted, the organizations had sophisticated and potentially effective safety programs and safety devices. In each case, the potential effectiveness of the safety devices was canceled out by non-technical factors. The concern, responsibility, and accountability for safety in an organization may be as important or more important than technology.

- *The operators of steam engines received most of the blame for accidents, not the designers or the technology.*

It is unfortunately very common to blame the operators for accidents when they have been put into a situation where human error is inevitable. This is as common today as it was a hundred years ago. And it is becoming a more serious problem as software engineers start to design human/machine interfaces

without adequate knowledge about human factors and without the benefit of decades of gradual improvement of designs through experience.

As an example, although it is almost universally believed that pilot errors account for the majority of aircraft accidents, an Air Force study of 681 in-flight emergencies showed 659 crew recoveries for equipment and maintenance deficiencies with only 10 pilot errors. Other aerospace studies show that about 80% of aircraft pilot-related accidents are due to poor training or neglect of human engineering in controls and instruments, not to stupidity or panic [18].

Humans are effective in emergencies because of their ability to analyze a situation and come up with novel solutions. Humans work well when they have a deep understanding, a sound model of the world, that they can use to predict the results of their actions. Operators sometimes find it necessary to violate the rules in order to accomplish their tasks or to prevent or mitigate the consequences of accidents. The disruption that often occurs during a job action when employees "work to rule" demonstrates how necessary flexibility is. In order to make decisions during emergencies, operators must have an understanding of the system they are controlling and must be given proper information in a usable format.

Three Mile Island is a classic example of the misattributing of an accident to operators and the use of hindsight to label operators' actions as erroneous. Operators are usually blamed for this accident although the accident sequence was initiated and compounded by equipment failure that was completely independent of operator action. Furthermore, the major errors of the operators could only have been seen after the fact; at the time, there was not enough information about what was going on in the plant to make better decisions. In fact, the events that occurred have been labelled as inevitable given the existing instrumentation [7]: They were a direct function of the electromechanical system design. For example, the computer was hours behind in printing out alarms and information although decisions had to be made in minutes, the instrumentation was unreadable under emergency conditions, and the wrong information was provided. Prior to the Three Mile Island accident, nuclear engineers took little interest in operator interface design. The Kemeny Commission's report on the accident concluded that the operator error was precipitated and compounded by basic flaws in system design [20].

The Vincennes (Iranian Airbus) incident is well known, but many other less-publicized accidents have occurred due to poor design of the human/computer interface. At one chemical plant in Britain, a computer printed a long list of alarms when a power failure occurred. The design team had assumed that in such a situation the operator would immediately trip (shutdown) the plant. Instead, the operator watched the computer print the list of alarms and wondered what to do. The operator should not bear the responsibility alone here; if any person is overloaded with too much information, they are most likely to do nothing while they try to understand the situation [21].

A basic understanding of human psychology and behavior is a prerequisite for user interface design that is commonly missing from software engineering education. A design, for example, that involves displaying data or instructions on a screen for an operator to check and to verify by pressing the enter button will, over time and after few errors are found, result in the operator getting into the habit of pressing the enter key multiple times in rapid succession. Most of us have fallen into this trap ourselves.

The solution is obvious. Software engineers must take human factors more seriously and human engineering experts must be involved in the design of safety-critical software interfaces.

- *The early steam engines had low standards of workmanship, and engineers lacked proper training and skills.*

Building safety-critical software requires special skills and knowledge on the part of both developers and management. Like any quickly developing technology, demand for qualified personnel has outstripped the supply, and appreciation of the skills and training necessary is often lacking.

Too often education in software engineering is behind the state-of-the-art, and it narrowly focuses on computer skills without providing training in basic engineering skills. All too typical is the man with a degree in nuclear engineering who told me that he builds software to control aircraft although he does not really understand basic aeronautical principles (and, I suspect, software engineering principles). People lacking in-depth knowledge of software engineering or the application area, and sometimes both, can be found building safety-critical software.

Many government standards in the U.S. require critical engineering projects to have at least one licensed Professional Engineer on their staff. System Safety Engineers have additional licensing requirements in many states. The standards do not usually

require that every engineer on a project have a Professional Engineering or Safety Engineering license; however, a license is required for those holding certain positions on the project such as lead engineer or system safety manager, along with requirements that they accept responsibility for assuring that the highest engineering standards and ethics are practiced. Nothing similar exists for any of the Software Engineers who are working on the same projects.

In his campaign against high-voltage electricity, Edison warned against the problems of poor workmanship and ignorance on the part of the majority of electrical contractors just as Watt had emphasized the personal moral responsibility of the engineer to ensure a safe and efficient steam engine and the culpability of the engineer in case of accidents [10]. If we in software engineering do not ourselves insist on establishing minimum levels of competency and safety, then the government will step in and do it for us. The public expects and has the right to expect that dangerous systems are built using the safest technology available.

Watt, Edison, and other inventors of the 18th century campaigned to raise professional skills because they realized the potential harm of their inventions in the wrong hands. They anticipated the need for higher standards of safety and precision in the engineering of new technological systems, and they initiated the process of raising professional standards [10]. Edison and Watt believed that "engineers had a responsibility to produce competent work, including the utmost in safety" [10]. Eventually professional societies developed that took over the role of establishing safety and competency standards.

Such standards and licensing requirements must be carefully composed. The extensive regulation of high-voltage electricity distribution in Great Britain has been blamed for its slow adoption and the lag in electrical development compared to the U.S. [26]. For example, regulations that set a minimum standard of insulation were stricter than was necessary and were blamed for the high cost of installation. But many British engineers argued that although the extensive regulation increased the cost, it also lessened the danger of fire and injury. As a group, British electrical engineers in the 1890's believed that lack of regulation in the U.S. had helped the development of the electrical industry at the cost of more accidents, which were "so common as to be part and parcel of the system" [26]. At the same time, British engineers were condemning Americans for their unsafe use and maintenance of steam boilers.

Just as overly strict regulations unnecessarily inhibited electrical technology development in Britain in the last century, so poorly-written standards can inhibit the development of computer technology. Worse, standards can inadvertently shift responsibility away from the manufacturers and developers to government agencies that have much less effective and direct control over the safety of the final product. And poorly written standards may have no effect or even increase risk.

Some current attempts to formulate software standards for critical systems equate safety and reliability (for example, the use of "integrity levels" which are usually just a pseudonym for reliability levels) or they define safety as the reliability of the safety protection devices (which is the prevailing definition in the nuclear power industry). While this approach to risk is common in reliability engineering, safety engineering has learned the hard way that highly reliable systems can be very dangerous while it is possible to design systems to be very safe even though they are unreliable. Limiting our standards to reliability concerns and enhancement only will not be effective against the large number of accidents that do not result from failures nor will they be effective against those accidents that do result from failures in systems or subsystems (like software) where ultra-high reliability cannot be achieved or guaranteed.

Safety engineers instead define safety in terms of hazards and attack the problem by looking for ways to eliminate or control hazards. Two approaches are possible: eliminating or minimizing the occurrence of hazards and controlling hazards once they occur in order to prevent injury or damage. As an example, if fire is the hazard of concern, the first approach would substitute nonflammable materials or eliminate or minimize the potential for a spark; in effect, the design becomes inherently safe and ensures that risk from fire is extremely low or non-existent. The second or protection system approach would instead rely on smoke detectors and sprinkler systems to detect and put out a fire after it starts; the risk then is dependent on the reliability of the protection device. Upstream approaches (hazard elimination or minimization) may result in a safer system but they may also require foregoing some benefits (e.g., reducing outputs or increasing development costs) or they may not be possible. Downstream approaches may require fewer design tradeoffs, but they may result in higher risk.

System safety analysis involves identifying and evaluating these tradeoffs in the early design stages of the system. Limiting our definitions and standards to the

use of protection devices effectively rules out the use of potentially more powerful approaches before they are even considered. Furthermore, relying on protection devices again limits our solutions to finding ways to build ultra-high reliability protection devices and ultra-high reliability software.

In our enthusiasm, we also do not want to impede progress by writing unachievable standards or inadvertently increase risk by implementing the wrong standards. As discussed earlier, we have not scientifically established the benefits and effectiveness of most of our software engineering techniques. Depending on a particular software engineering methodology to assure safety by assuming it will produce error-free or ultra-high reliability software is dangerous. And as the technology progresses, standards that require the use of specific approaches often lag behind. Manufacturers may feel no ethical or legal duty to go beyond what is required in the standard.

Moreover, manufacturers or those who will personally benefit financially from particular techniques being included or not included in the standards sometimes play a dominant role in the drafting process. The result may be watered down requirements or the recommendation of techniques with more commercial than technical value.

The alternative is to construct flexible standards specifying general criteria for acceptability of a methodology instead of a specific methodology and ensuring that those building safety-critical software have the competency and personal responsibility to use the best approaches available at the time and for the particular project characteristics.

As Edison argued with respect to electricity, increased government regulation of our technology may not be to anyone's benefit; but it is inevitable unless we, as the technology's developers and users, take the steps necessary to ensure safety in the devices that are constructed and technical competence in those that construct them.

Acknowledgements

Several people provided helpful comments on earlier drafts of this paper including Daniel Berry, John Gannon, Susan Gerhart, David Notkin, David Parnas, Jon Reese, John Rushby, and Elaine Weyuker. It should not be assumed, however, that they necessarily agree with the points made in the paper.

References

[1] Aitken, A. Fault Analysis, in A. E. Green (ed.), *High Risk Safety Technology*, New York: John Wiley & Sons, 1982

[2] Archinoff, G.H., Hohendorf, R.J., Wassyng, A., Quigley, B., and Borsch., M.R. Verification of the Shutdown System Software at the Darlington Nuclear Generating Station, *Proc. Int. Conf. on Control and Instrumentation in Nuclear Installations*, Glawgow, U.K., May 1990.

[3] Bollinger, T. and McGowan, C. A Critical Look at Software Capability Evaluations, *IEEE Software*, July 1991, pp. 25–41.

[4] Bowman, W.C., Archinoff, G.H., Raina, V.M., Tremaine, D.R., and Leveson, N.G. An Application of Fault Tree Analysis to Safety Critical Software at Ontario Hydro, *Conf. on Probabilistic Safety Assessment and Management (PSAM)*, Beverly Hills, April 1991.

[5] Briggs, A. *The Power of Steam*, Chicago: The University of Chicago Press, 1982.

[6] Brilliant, S.S., Knight, J.C., and Leveson, N.G. Analysis of Faults in an N-Version Software Experiment, *IEEE Trans. on Software Engineering*, Vol. SE-16, No. 2, February 1990, pp. 238–247.

[7] Brookes, M.J. Human Factors in the Design and Operation of Reactor Safety Systems, in D.L. Sills, C.P. Wolf, and V. Shelanski (eds.), *Accident at Three Mile Island: The Human Dimensions*, Boulder, Colorado: Westview Press, 1982.

[8] Brooks, F.P. No Silver Bullet: Essence and Accidents of Software Engineering. *IEEE Computer*, April 1987, pp. 10–19.

[9] Burke, J.G. Bursting Boilers and the Federal Power, *Technology and Culture*, Vol. VII, No. 1, Winter 1966, pp. 1–23.

[10] Cameron, R. and Millard, A.J. *Technology Assessment: A Historical Approach*, Dubuque, Iowa: Kendall/Hunt Publishing Company, 1985.

[11] Dickens, Charles. *Household Words, 1851*, in Stone, Harry (ed.), *Uncollected Writings from Household Words, 1850–1859*, Bloomington: Indiana University Press, 1968.

[12] Dickinson, H.W. *A Short History of the Steam Engine*, London: Frank Cass & Co. Ltd., 1963.

[13] Eckhardt, D.E., and Lee, L.D. A Theoretical Basis for the Analysis of Multiversion Software Subject to Coincident Errors, *IEEE Trans. on Software Engineering*, Vol. SE-11, No. 12, December 1985, pp. 1511–1516.

[14] Eckhardt, D.E., Caglayan, A.K., Knight, J.C., Lee, L.D., McAllister, D.F., and Vouk, M.A. An Experimental Evaluation of Software Redundancy as a Strategy for Improving Reliability, *IEEE Trans. on Software Engineering*, Vol. SE-17, No. 7, July 1991, pp. 692–702.

[15] Farey, J. *A Treatise on the Steam Engine: Historical, Practical, and Description*, London: Longman, Rees, Orme, Brown, and Green, 1827.

[16] Goodenough, J. B. and Gerhart, S. Toward a Theory of Test Data Selection, *IEEE Transactions on Software Engineering*, Vol. SE-1, No. 2, June 1975.

[17] Hills, R.L. *Power from Steam: A History of the Stationary Steam Engine*, Cambridge: Cambridge University Press, 1989.

[18] Johnson, W.G. *MORT: Safety Assurance Systems*, New York: Marcel Dekker, Inc., 1980.

[19] Josephson, M. *Edison*, London: Eyre and Spottiswoode, 1961.

[20] Kemeny, John G. The Need for Change: The Legacy of Three Mile Island. *Report of the President's Commission on Three Mile Island*, New York: Pergamon Press, 1979.

[21] Kletz, T. Wise After the Event, *Control and Instrumentation*, Vol. 20, No. 10, October 1988, pp. 57–59.

[22] Knight, J.C. and Leveson, N.G. An Experimental Evaluation of the Assumption of Independence in Multiversion Programming, *IEEE Trans. on Software Engineering*, Vol. SE-12, No. 1, January 1986, pp. 96–109.

[23] Knight, J.C. and Leveson, N.G. A Reply to the Criticisms of the Knight and Leveson Experiment, *Software Engineering Notes*, January, 1990.

[24] Leveson, N.G. and Turner, C.S. The Story Behind the Therac-25 Accidents: A Computer-Related Accident Investigation, submitted for publication.

[25] Millard, A.J. *Edison and the Business of Innovation*, Baltimore: Johns Hopkins University Press, 1990.

[26] Millard, A.J. *A Technological Lag: Diffusion of Electrical Technology in England 1879–1914*, New York: Garland Publishers, 1987.

[27] Parnas, D.L. Software Aspects of Strategic Defense Systems. *Communications of the ACM*, Vol. 28, No. 12, December 1985, pp. 1326–1335.

[28] Parnas, D.L. Why Engineers Should Not Use Artificial Intelligence. *Proceedings of the CIPS Edmonton '87 Conference*, Edmonton, Alberta, November 16-19, 1987, published in J. Schaeffer and L. Stewart (eds.), *Intelligence Integration*, Dept. of Computing Science, University of Alberta, p. 39–42.

[29] Passer, H. *The Electrical Manufacturers*, Cambridge, Mass.: Harvard University Press, 1953.

[30] Pursell, C.H. *Early Stationary Steam Engines in America*, Washington, D.C.: Smithsonian Institution Press, 1969.

[31] Rasmussen, J. Cognitive Control and Human Error Mechanisms, in J. Rasmussen, K. Duncan, and J. Leplat (eds.), *New Technology and Human Error*, New York: John Wiley & Sons, 1987.

[32] Robinson, E. and Musson, A.E. *James Watt and the Steam Revolution*, New York: Augustus M. Kelley, Publishers, 1969.

[33] Ruckelshaus, W.D. Risk, Science, and Democracy, in T.S. Glickman and M. Gough, *Readings in Risk*. Washington, D.C.: Resources for the Future, 1990.

[34] Scott, R.K., Gault, J.W., and McAllister, D.F. Fault-Tolerant Software Reliability Modeling, *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 5, May 1987, pp. 582–592.

[35] Watts, S. Computer Watch on Nuclear Plant Raises Safety Fears, *London Independent*, Sunday, Oct. 13, 1991.

[36] Weil, V. The Browns Ferry Case, in M. Curd and L. May (eds.) *Professional Responsibility for Harmful Actions*, Dubuque, Iowa: Kendall Hunt, 1984.

[37] Weyuker, E.J. Axiomatizing Software Test Data Adequacy, *IEEE Trans. on Software Engineering*, Vol. SE-12, No. 12, Dec 1986, pp. 1128–1138.

[38] Weyuker, E.J., Weiss, S., and Hamlet, D. Comparison of Program Testing Strategies, *Proceedings of the Fourth Symposium on Software Testing, Analysis and Verification (TAV4)*, Victoria, B.C., Canada, Oct 1991, pp. 1–10.