Proving the Value of Formal Methods

Gerard J. Holzmann
AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

The record of successful applications of formal verification techniques is slowly growing. Our ultimate aim, however, is not to perform small pilot projects that show that verification is sometimes feasible in an industrial setting; our aim must be to integrate verification techniques into the software design cycle as a non−negotiable part of quality control.
We take a snapshot of the state of the art in formal verification, and, for inspiration, compare it with other points in history where new technology threatened to replace old technology, only to discover how resilient an established practice can be. To keep a mild sense of perspective, we will also briefly consider how badly mistaken some of the advocates of new technology have sometimes been.

Invited paper for FORTE94, 7th Int. Conference on Formal Description Techniques. To be held in Bern, Switzerland, October 1994.

The difference between a thing that can break
and a thing that can't break is that when the thing
that can't break breaks then it can't be fixed.
(Hitchhiker's Guide to the Galaxy, Book 5)

1. *INTRODUCTION*

Despite a number of projects that appear to have demonstrated successfully that the adoption of formal design techniques in a mainstream industrial environment is feasible and beneficial, e.g., [Ch91],[Ru92],[Ho94], in only few cases have such projects led to a permanent change in the design process that they targeted. How, then, can we prove the value of formal methods in such a way that also this second step is taken? To do so, we will have to accomplish two separate objectives. First, we will have to show in a rigorous scientific way (i.e., based on objective measurements and hard data) that the introduction of new formalism can indeed benefit a systems designer. Second, we have to show that adequate formalisms exist today that are ready to be applied in an industrial design environment. Both these objectives have so far proven to be hard to realize. In the following we will examine them in a little more detail.

1.1 *The First Objective: Rationale*

The first objective would appear to be the easier of the two. It is, however, much easier to reason abstractly about the rationale for formal methods, then it is to demonstrate its correctness in a rigorous scientific manner.

Clearly, the basic notion that a sound design has to be based on some type of formal reasoning is not very new. Epictetus (AD 50−c.130), for instance, used the following example in *The Discourses*, to make that point:

> When one of those who were present said, ''Persuade me that logic is necessary,'' he replied: ''Do you wish me to prove this to you?'' The answer was, ''Yes.'' . . . ''How then will you know if I am not cheating you by argument?''

Some form of logic inevitably underlies any design. The main purpose of a formal design method is not to introduce logic for its own sake, but to make the existing logic visible, so that its soundness can be checked. A formal design technique introduces the minimum of formalism that is sufficient to *expose* the logic of the reasoning process that underlies a design, and thus opens up this process to formal scrutiny.

But the above is only an abstract justification that appeals mostly to those of us who are already advocating the use of formal methods. It does not appeal directly to a practitioner, and it fails to provide objective data to support the thesis that formal methods are useful. If presented and justified only informally, the target users will perceive the added formality as, indeed, mere ‘‘formality:’’ extra *tasks* to be performed without visible benefit. John Chaves, a regular participant in formal methods projects at AT&T Bell Laboratories, observed that new technology has a much greater chance of being adopted if it is perceived as a *service*, with measurable benefits, rather than a *task*.

At the very minimum we must be able to make the formal method appear to be a service to the user, with a clear and immediate benefit. Even this is often much harder than it seems. It is the sentiment expressed by Dijkstra, when he wrote, in [Dij76]:

> I found this effort at clarification to my amazement (and annoyance) repeatedly rewarded by the accusation that ‘‘I had made programming difficult.’’ But the difficulty has always been there, and only by making it visible can we hope to become able to design programs with a high confidence level . . .

If we stay with the informal argument a little longer, we could say that the formal method is a *service* that secures the integrity of the design process, and perhaps more convincingly, the integrity of the designer. In return for satisfying the requirements of a formal method, the designer gains the capability to discover the mistakes that have slipped into the design, and gains the opportunity to repair them before they can tarnish his reputation. As Epictetus wrote, also in *The Discourses*:

> For if any one shall show [that a mistake was made], a man will himself withdraw from that which he does; but so long as you do not show this, do not be surprised if a man persists in his practice; for having the appearance of doing right, he does what he does.

This still does not, however, solve the problem in practice. Being very human, most designers (and researchers) tend to deny the fact that they can err. This brings us to our original objective: to show in a rigorous and scientific way that the introduction of a formal method is beneficial.

In the *NewCoRe* project, a pilot formal methods project we performed at AT&T between 1990 and 1992, our goal was to provide the much needed hard scientific data, [Ch1], [Ho94]. As part of this experiment, a single development project was pursued in two different ways, by two different groups in parallel. The first team used a traditional design method; the second (smaller) team applied formal design and verification techniques. Our aim was to compare the two parallel designs at the end of two years with conventional lab testing techniques, and then determine objectively which design constituted the better product.

The reality the NewCoRe team had to confront, however, was that as soon as it started discovering the first series of logical inconsistencies in the project requirements (after about six months), these errors could not reasonably be kept secret from the parallel team. Our wish for a double blind scientific experiment therefore had to be sacrificied to the realities of a competitive industry. The early success of the NewCoRe project in this case meant that it could no longer prove unambiguously and quantitatively exactly how much better a formal method can be. (Some more details on the project follow in section 2.3.)

1.2 *Opponents*

Formal methods are often opposed by practitioners on purely pragmatic grounds, cf. also [Ha90] and [BoHi94]. The following is from an opponent of formal methods, who shall remain anonymous:

> A formal design method is often overkill. Only a small portion of the errors found by an expensive formal design process would ultimately show up in practice.

And similarly, quoted verbatim from a cautious manager of a formal methods project:

> The paradigm of ‘‘zero faults the first time through’’ is often not as good as an iterative approach to development [i.e., based on testing].

Issues such as these can effectively block the permanent adoption of a formal method, and will need to be addressed seriously. Unfortunately, there are no one−sentence answers.

The first issue is one of economics: can an industry afford to build a higher quality product by retooling the design cycle for a formal method. Any change in an industrial design process involving more than a few engineers is costly, and will be weighed on its immediate merits. The chances of adoption are far greater if there is a measurable interval in which the investment pays for itself. There are no statistics on the types of faults that could be prevented from reaching a delivered product and causing damage, and thus it can be hard to make such a case.

The second issue may be easier to address, merely by properly defining the difference between traditional testing and a design method based on formal verification. Traditional product testing is based on the premise that the design starts with a trusted set of requirements, and is implemented via an untrusted process. Testing is meant to verify that the final product conforms to the original requirements. An example is a craftsperson building a cabinet. When the work is complete, (s)he will test that all the drawers open smoothly, and close tightly, and that the cabinet doesn't wobble. There is never any question here that the original design plan for the cabinet as such is in order, and would lead to a proper product if implemented well.

In formal software verification a quite distinct problem is tackled. Unlike the cabinet−maker, we cannot trust our design requirements: they are often incomplete, ambiguous, are even self−contradictory. In a traditional design cycle one expects to discover the ambiguities in the original requirements with testing methods that are not meant, nor capable of, reliably exposing logical inconsistencies in the requirements themselves. In software verification, traditional testing offers too little and it offers it too late in the design cycle (i.e., after the developer has already committed to the implementation of potentially inconsistent requirements).

1.3 *The Second Objective: Methodology*

It is perhaps unfortunate that as long as computers have existed, there have been people who tried to convince their peers that they were powerful enough for deriving or proving significant facts. The reasoning is always persuasive, because in effect there exists no real alternative to mechanical verification. As noted in [ELWW72, p. 120, 143]:

> Perhaps the primary argument against the informal proof technique is that it is as fallible as the person carrying out the proof. . . .
> It is our feeling that a mechanical verifier is the only foolproof approach.
> . . . informal proofs of large programs cannot be handled reliably by a human being, and since the art of informal program proving is not likely to improve, mechanical (or mechanically assisted) verification is the only approach for large programs.

But placing too great a faith in the potential of mechanized verification makes an easy target for the sceptic, since there will always be facts that cannot be proven or disproven by any means, and many of them easily turn up in practice. Edsger Dijkstra wrote, almost twenty years ago [Dij75]:

> We see automatic theorem provers proving toy theorems, we see automatic program verifiers verifying toy programs and one observed the honest expectations that with faster machines with lots of concurrent processing, the life−size problems will come within reach as well.

It could as easily have been written today, even now that our computing machinery runs at least three orders of magnitude faster than in 1975.

It is probably fair to say that we have not yet been able to identify the subdomain where formal methods can be applied without second thoughts, routinely, and with predictable benefit. As Colin West noted in [We93]:

> . . . the FDT's that were available promised the ability to produce technically superior specifications, but failed to satisfy time, resource and ease−of−use constraints.

Overstating the case for formal methods, in this context, may do as much damage as underestimating it. As [BoHi94] phrased it:

> The excessive optimism of the attitude that everything important is provable helps to explain the excessive pessimism of the attitude that nothing important is provable.

Still, there is of course no need to be depressed about the future of our field. Especially in the last few years much progress has been made in the construction of efficient automated verification systems, and in their application to problems of practical significance (e.g., [Ch91], [Ho94]). Where only recently research focused on efficient methods for proving simple properties such as absence of deadlock, a state−of−the−art verification system today handles correctness requirements formalized in temporal logic, and uses sophisticated complexity management techniques to deliver optimal performance. Yet, it would be folly to believe that these tools can be applied blindly to any design problem and deliver instant rewards. System modeling, the formalization of correctness requirements, model verification, and, where necessary, model reduction, are skills that must be learned. Fortunately, many universities are leading the way with courses on formal verification and design techniques. The new skills will therefore inevitably reach their target.

2. *STATE OF THE ART*

For the purposes of this paper the discussion here is restricted to a small sub−field of the general area that is covered by formal methods: the application of automated verification techniques to the proof of logical correctness requirements for concurrent systems. Traditionally, then, there are two approaches to this problem, both first initiated in the early 1980s. The first is usually termed *model checking*, the second is known as *on−the−fly verification*.

Model checking was pioneered by Ed Clarke and his students at Carnegie−Mellon University, e.g., [ClEm81], and by Joseph Sifakis and his students at the University of Grenoble in France, e.g., [Qu82]. It is essentially a two−phase verification process. In the first phase, the reachable state space of a concurrent system is computed and the essential features of it are encoded into a data−structure that is stored in a disk−file, or held in main memory. In the second pass the validity of a correctness requirement, often formalized in a temporal logic formula, is verified for the given global state space structure.

On−the−fly verification has its roots in the mid−seventies, e.g. [Su75], but only began in earnest with the work of Colin West at IBM in the early eighties, e.g., [RuWe82]. The effort here was to prove every property of interest for a concurrent system in a single pass of the state space exploration algorithm. There were many parallel attempts to build verification systems with this property, e.g. [Ho81]. It was not until much later that the first on−the−fly verification systems were built that could verify also temporal logic requirements. As far as we know, the first efficient implementation of such a system was SPIN, which is detailed in [Ho91] and incorporates ideas on temporal logic verification from Costas Courcoubetis and Piere Wolper.

2.1 *Application Domains*

The model checking techniques have proven their strength primarily in their application to hardware circuit verification problems. On−the−fly verification techniques, on the other hand, have their strength primarily in software verification. There are several fundamental differences between these two areas of application, that have led to a quite distinct line of development in the area of model checking and on−the−fly methods. First, hardware systems are typically synchronous, and exhibit a regular, often repetitive, structure (e.g., counters, registers, arrays), that can be exploited successfully by customized verification algorithms. Software systems, such as communication protocols or distributed operating systems, are always (at least logically) asynchronous, not driven by a global clock, and they do not exhibit the same amount of regularity or structure.

The difference in application domain may explain why the application of symbolic data storage techniques, such as binary decision diagrams (BDDs), or symmetry reduction techniques have been more successful when applied in model checking algorithms than when they are applied in on−the−fly verification algorithms. Similarly, it can explain why the application of partial order theories could be shown to be beneficial in on−the−fly systems modeling asynchronous process interleavings, [HoPel94], but (as yet) not in model checking systems modeling synchronous hardware.

One of the advantages of the on−the−fly verification technique is that there is no strict need to construct an accurate global representation of the system behavior as a precondition for performing verifications. The focus of the research in these systems has therefore been to find efficient algorithms that construct the smallest possible portion of the global system that suffices to prove or disprove a temporal logic requirement. It is often sufficient in these cases to compute only an approximate representation of the global state graph, without sacrificing proof accuracy in any way (i.e., by applying coverage preserving reduction techniques).

2.2 *Changes in Limits*

Although the limits have been moved substantially in the last ten to fifteen years, computational complexity still lies at the very heart of the verification problem, and it is likely to remain there for the foreseeable future. Curiously, the improvements in cpu speed of the last decade (roughly three orders of magnitude) have not been matched by an equivalent increase of the available main memory size (roughly one order of magnitude less). The immediate implication of this phenomenon is that what used to be the main bottleneck preventing exhaustive verification of large systems (cpu−time) has disappeared, but has been replaced with another. Today, full−scale verification, both in model checkers and in on−the−fly systems, is primarily limited by the memory requirements of the algorithms. A 100 Mbyte state space can be exhaustively explored in minutes, but this much memory does not always suffice to perform complete verification of larger systems. The challenge in automated verification today, therefore, is to increase the memory efficiency of the search algorithms, and simultaneously to find effective ways to reduce the complexity of validation models *before* they are subjected to automated verification.

2.3 *Some Recent Application Successes*

We first started applying on−the−fly verification techniques to routine development projects at AT&T in 1988 [HoPa89]. Based on the positive results of this first application, it was decided to prepare a custom−made version of a verification system for the AT&T extension of the ITU (formerly CCITT) language SDL (Specification and Description Language), which was then the standard for describing the protocol aspects of the AT&T switching systems. This program, called SDLVALID, was completed in 1990, and first applied in the NewCoRe project, mentioned earlier at the end of section 1.1.

The five people who made up the NewCoRe team built a series of validation models for a new ISDN protocol that was in development, the largest model containing 7,500 lines of non−comment SDL source. A total of 145 correctness properties were formalized in temporal logic and verified with SDLVALID. Over a two−year period close to 10,000 verification runs were performed; an average of more than 100 verification runs per week.

The NewCoRe team uncovered a total of 112 serious design errors in the design requirements, by automated verification runs. These errors could be prevented from reaching the product, without requiring field−tests. The NewCoRe effort also successfully demonstrated that close to 55% of the original requirements for one of the designs covered were logically inconsistent, and thus could not have been realized adequately by any implementation. More details on the NewCoRe project can be found in [Ch91] and in [Ho94].

2.4 *Design for Verifiability*

The particular approach to formal design methods that we are pursuing is best summarized in the following principles.

1. A good design method should allow one to
   a. Discriminate between requirements and specifications
   b. Built engineering prototypes of a design
   c. Use these prototypes to predict the essential characteristics of the design
2. A good design method, therefore, minimally requires
   a. An unambiguous notation to specify system behavior
   b. An independent formalism to specify the logical requirements on that behavior
   c. A software tool to verify that the requirements are satisfied for the behavior as specified.

Very few of the existing formal methods standards can fully satisfy these requirements. The specification language SDL, for instance, has no facility for items 1a or 2b, and therefore in its basic form precludes items 1c or 2c. In our applications we therefore first extended the basic language to provide the minimal features necessary for a formal method in the above sense, cf. [HoPa89], [Ch91], [Ho94]. Similarly, few of the existing FDT's were developed with an eye towards the feasibility of 2c. The basic language definition of SDL, to remain with this example, stipulates that all message channels between processes are unbounded. This precise feature makes all correctness requirements formally undecidable, and therefore in our applications too we had to modify this definition and require the use of only explicitly bounded channels.

But modifying an existing FDT to allow for formal verification remains patchwork. Largely as an experiment, the validation modeling language PROMELA [H91] was designed explicitly with efficient verifiability in mind. This PROtocol MEta LAnguage is specifically not meant to represent implementation detail. Being experimental, it undeniably lacks many of the features that would be required for its use as a general FDT, and therefore it can not realistically compete with these. The main limitation of PROMELA, however, is also its strength: it tries to do only one thing (validation modeling) and do it well.

In this context then, the main purpose of a formal method is to enable (empower) the designer to represent design decisions conveniently, accurately, and unambiguously, and, most importantly to enable the designer to verify the logical consistency of those design decisions long *before* the implementation stage is reached. *Design for verifiability*, then, is the new paradigm of design that we hope the formal methods community can advance as the single most important alternative to current software engineering practice.

3. *HISTORICAL PERSPECTIVE*

It is tempting to think that in formal verification we are witnessing the birth of a new methodology of software design. So much of our environment relies on the correct working of software that it is almost inconceivable that we have managed to do without any type of formal method for so long. The very idea that algorithms or protocols can be endorsed by major standardization institutes without any form of formal verification may some day make an unbelievable anecdote of our time. Many of these standards are readily proven incomplete or faulty, simply by running any one of the readily available verifiers.*

We can safely leave it undecided here whether formal methods will ultimately replace the traditional style of software design, and for our own amusement merely assume that this will be so. Fortunately, it is much easier to look back in time than it is to look forward. We can therefore look at points in recent history where people rightfully or mistakenly believed they were in a similar situation. The most appealing examples, then, are at the innovations that we now accept as a quintessential part of our environment. None of them were unopposed, and in some cases the opposition against their adoption caused a delay that lasted for a generation or longer. As examples, we will look at the introduction of the first mechanical computing devices (generously called ‘computers’ here) by Charles Babbage in ca. 1825, the introduction of the electrical telegraph by Samuel Morse and others in ca. 1837, and the invention of the telephone by Alexander Graham Bell and others in ca. 1876. More details on most of these examples can be found in [HoPehr95].

3.1 *Computers*

One of the motivations for Charles Babbage’s work on his analytical engine was to rule out the possibility of human error in repetitive computational tasks, such as the compilation of astronomical and logarithmical tables. Until (relatively speaking) fairly recently such tables were always produced by human computers, and inevitably they contained mistakes. This is how Babbage described how he got the idea to built his first engine, in his autobiography, see [Mo61, p. xiv]:

> Herschel brought in some calculations done by [human] computers for the Astronomical Society. In the course of their tedious checking Herschel and Babbage found a number of errors, and at one point Babbage said ‘‘I wish to God these calculations had been executed by steam.’’

When Babbage had succeeded in building his first set of mechanical devices that could perform these tasks mechanically, he had a very hard time convincing the official institutions, such as the Royal Astronomical Society, to change their custom and to use his machines instead of the traditional human computers. Babbage wrote, [Mo61, p. 69]:

> The great importance of having accurate tables is admitted by all who understand their uses; but the multitude of errors really occurring is comparatively little known. . . .
> . . . I have no intention of imputing the slightest blame to the Astronomer Royal, who, like other men, cannot avoid submitting to inevitable fate. The only circumstance which is really extraordinary is that, when it was demonstrated that all tables are capable of being computed by machinery, and even when a machine existed which computed certain tables, that the Astronomer Royal did not become the most enthusiastic supporter of an instrument which could render such invaluable service to his own science.

*) E.g., the SPIN system from [Ho91], which is available by anonymous FTP from the server `netlib.att.com` (directory `/netlib/spin`).

3.2 *Telegraphs*

The first patent on an electrical telegraph was issued on 12 June 1837 to William Cooke and Charles Wheatstone in England. It was titled: *Method of Giving Signals and Sounding Alarums at Distant Places by Means of Electric Currents Transmitted through Metallic Circuits*. But electrical telegraphs were being studied by many at this time, especially, of course, also by Samuel Morse. Morse gave his first public demonstration of a so−called ‘‘pendulum’’ telegraph at New York University on 3 September 1837, with Alfred Vail in the audience. Vail was a student at N.Y.U. at the time, and joined Morse as a collaborator after this demonstration. The Morse code, and the final version of the Morse– Vail telegraph both date from 1838, and unofficially the first message sent with this device by Vail, in Morse’s workshop in Speedwell, New Jersey, was ‘‘A patient waiter is no loser.’’

Cooke and Wheatstone’s device was based on the deflection of magnetic needles by an electric current. In retrospect, it is easy to see the advantages of the Morse– Vail system over this device: it was simpler, it (inadvertently at first) made the signals both audible and visible, and it could easily be modified to leave a permanent record of the transmissions on strips of paper. Nonetheless, while Cooke and Wheatstone’s device found almost immediate practical application in 1837 on a stretch of railway between Euston and Camden Town in England, Morse was unable to convince anyone to use his device for almost 7 years.

In 1837 the United States Congress was considering a plan for the construction of a telegraphic connection between Baltimore and New Orleans, based on optical semaphores (then the standard method for long−distance communications, see [Ho&Pehr94]). Morse submitted a proposal to Congress to use his electrical telegraph instead. Congress, however, could not be persuaded that it was more attractive to use a technique that required not only the construction of hundreds of telegraph stations (as the optical semaphore also required) but also the stringing of expensive copper wires between them. So, Morse’s proposal was rejected.

The next year, in 1838, Morse visited France to try his luck there. He demonstrated his telegraph to the French Academy of Sciences, and bid on a contract to install an electrical telegraph line along the railway from Paris to Saint−Germain. In February 1839 Morse left France again, unsuccessful also in this bid. The French government decided that the installation and exploitation of a telegraph network was too important to be trusted to a private individual (and a foreigner at that). It took until 1844 before one of Morse’s proposals was finally accepted, and he could speak his famous official ‘first’ message ‘‘What hath God wrought?’’ at the inauguration of the first American electrical telegraph line connecting Washington to Baltimore (24 May 1844).

In France it took still a little longer to switch to electrical telegraphs. In this case it was a prime example of a country being disadvantaged by its leading role. France was the first country in the world to institute a State Telegraph. This happened on 26 July 1793 (not a typo). The telegraph that was used to construct the first ever nation−wide telegraph network was based on an optical semaphore−telegraph invented by Claude Chappe (see [Ho&Pehr95]). By 1840, this network was well established and counted nearly 500 stations, connecting all the main cities of France. The electrical telegraph, therefore, first and foremost had to prove its superiority over the ‘‘traditional’’ optical telegraphs. Dr. Jules Guyot took it upon himself to make a study the new devices, traveling to England to see them in operation first−hand. Upon his return to France he published an article in the *Courier Francais* that appeared 5 July 1841. It argued convincingly:

> Every sensible person will agree that a single man in a single day could, without interference, cut all the electrical wires terminating in Paris; it is obvious that a single man could sever, in ten places, in the course of a day, the electrical wires of a particular line of communication, without being stopped or even recognized.

He concluded:

> . . . it [the electrical telegraph] cannot seriously be considered suitable for the needs of the government, and the day is not far distant when this truth will be clearly demonstrated.

The first electrical telegraph lines were installed in France in 1845, but they were not Morse telegraphs. The device that was developed for the new lines, was designed by Alphons Foy, chief administrator of the optical telegraph network at the time, and Louis Breguet, and engineer. It carefully duplicated the positions of the optical Chappe semaphore on the controls of the electrical telegraph (backward compatibility). It took until 1855 before finally these devices were abandoned in favor of the simpler Morse telegraphs.

3.3 *Telephones*

By the 1870s the electrical telegraph was well established. A problem that had been signaled, and that was studied by many, was that a single telegraph wire could only be used for a single communication at a time. If a method could be devised to multiplex Morse signals for two or more connections onto a single wire, the telegraph business could save substantial amounts of money. There was a promise of great riches to the first person who could find a way to accomplish this.

Two of the people who were pursuing this problem were Elisha Gray and Alexander Graham Bell. Gray (1835–1901) was an established inventor with several patents in telegraphy to his name. In comparison, Bell (1847–1922) was an amateur, without formal training in electricity, and with only a scanty knowledge of telegraphy. Both Bell and Gray were working on prototype systems that could transmit a small number of frequencies over a telegraphic wire. Each frequency could, in principle, be used for a separate channel in a multiplexed link. Both Bell and Gray realized at one point or another that if you could transmit sufficiently many tones, you could also transmit sound. Here is for instance what a reporter wrote in The New York Times, May 1874, after witnessing one of the public demonstrations given by Elisha Gray, as quoted in [Ho81]:

> . . . in time [telegraph] operators will transmit the sound of their own voices over the wire, and talk with one another instead of telegraphing.

Gray consider this possibility an amusing side−application, at best, well divorced from the real money−making application: a multiplexing telegraph. Bell, on the other hand, decided that the transmission of sound or voices was the more seminal step, and he dropped the pursuit of a multiplexing telegraph entirely. Gray, of course, couldn’t be happier. In a letter to his patent attorney, Elisha Gray wrote in 1875:

> Bell seems to be spending all his energies in [the] talking telegraph. While this is very interesting scientifically it has no commercial value at present, for [the telegraph industry] can do more business over a line by methods already in use than by that system.

A few months later, Gray nonetheless let himself be persuaded by his attorney to file a patent claim on his idea As best as can be determined by coincidence, that same day Bell also filed a patent claim for the transmission of sound over telegraph wires, titled *Improvements in Telegraphy*. After the interference on the claims was discovered, Gray readily granted precedence to Bell, stating in a letter to Bell that though he had had the same idea, he had never attempted to reduce it to practice, and therefore could not rightfully lay claim to the invention. This may seem like a giant mistake on the part of Gray, and we can safely assume that his attorney had recommended against this admission, but things looked very different at this time. The odds seemed to be very much in Gray’s favor.

When, for instance, Bell offered his telephone patent to the telegraph company Western Union in 1876 for $100,000 he was turned down. Instead, Western Union bought the rights to Elisha Gray’s patents on multiplexing telegraphs. The telegraph industry, like Gray and many others, could not see what possible purpose there could be in having telegraph operators talk to each other over the wires. As we know, Bell went in business for himself and could prove the telegraph industry wrong in due time.

3.4 *Windmills*

The remarkable ability of the human mind to resist technological improvements is, of course, not limited to recent history. For proper measure, we therefore look at one more example, from a few hundred years earlier: improvements in windmill design.

In 1593 the Dutch farmer Cornelis Corneliszn invented a new transmission and gear−mechanism for windmills. The mechanism, the reverse of a modern crankshaft, could translate the rotation of a windmill’s main axle efficiently into a lateral movement that could drive, for instance, a powerful saw. Corneliszn received a patent on 15 December 1593, that protected his invention for a period of twelve years. The innovation led to a small revolution in the way that a saw mill was organized, mechanizing much of the duties that could only be performed by hand, and at a much slower pace, until then. The improvement was also timed well, with the great demand for wooden ships that was triggered by the exploits of the East and West Indian Companies in Holland in the beginning of the 17th century.

By 1630 there were 368 conventional mills in Holland (used for milling grain etc.), and 86 wind−driven sawmills. The 368 conventional mills were spread evenly over all districts, including the capital city

Amsterdam. But, curiously, not a single wind−driven sawmill could be found inside the gates of Amsterdam. The reason? The Union of Wood Millers successfully fought the mechanization of saw mills, considering it a serious threat to their profession. Although they couldn’t stop their construction elsewhere, the union had succeeded in banning the new mills from Amsterdam entirely.

It took almost a century for the ban to be broken. By 1730, finally, there were 78 wind−driven sawmills in Amsterdam, more than in any other city in Holland.

4. *IN CONCLUSION*

With the benefit of hindsight it is of course all too easy to say who was right and who was wrong to reject a new idea. When the events evolve in real−time, the issues are often much less clear. As just one example of someone who was rightfully denied the chance to realize his vision, let use consider the inventor of a telephone predating the ones of Bell and Gray by 80 years: the German professor Gottfried Huth (1763– 1818). In 1796 Huth published a translation of a work on acoustics from 1763 by the French physicist J. H. Lambert. Huth gave his book the title *A treatise concerning some acoustic instruments and the use of the speaking tube in telegraphy*. In his own additions to the translation, Huth criticized the Chappe optical telegraphs used in France at this time for being obviously flawed, since they could not be used in bad weather or at night. Huth improvement was to construct telegraph towers at roughly 5km intervals, on which men were positioned equipped with megaphones (‘‘speaking tubes,’’). These operators could shout messages to each other, be it sunshine or rain, day or night. Realizing the seminal importance of his idea, Huth knew he needed an original name for his invention, to set it apart from all others. He wrote:

> The fundamental difference therefore deserves, and will ultimately require, a different name for the telegraph based on speaking tubes. But what name would be more appropriate than the word derived from the Greek: ‘‘Telephon,’’ or far−speaker [German: *Fernsprecher*].

As we know, Huth did not get his way. Many others shared this fate with him. One of them, Johann Bergstrasser (1732– 1788), started publishing his first works on telegraphy in 1785. He was never very specific in the type of device he had in mind, although he clearly wanted to distinguish his work from his competitors:

> I do not use speaking tubes, and even less electricity or magnets; also no wires or tubes, like Mr. Gauthen [Gauthey]. And still, part of my apparatus can be placed in a closed room, anywhere in a large palace, wherever desired; . . . without it being necessary that the correspondents can see each other, or perform any of the operations themselves . . .

and elsewhere:

> I am completely convinced of an inexpensive construction, and of the feasibility of the scheme. Experiments cannot cost much; they can, however, only be performed in the field; such as the land of a wealthy nobleman who can spend a few thousand thaler on a useful speculation.

When Bergstrasser failed to convince any wealthy noblemen, he became increasingly annoyed at those who were more successful. When, for instance, Claude Chappe had successfully obtained support for the construction of the first optical telegraph lines in France, in 1793, Bergstrasser wrote in anger:

> That my proposal to connect Berlin via Leipzig, Magdeburg, Hanover to Hamburg has never been executed, like the proposal of the gentlemen Chappe to connect Paris to Lille, is not the fault of my method.

Some were even less gracious. When the British lieutenant John Macdonald failed, after 30 years of trying, to obtain support from the British Admiralty for his plans to improve the signaling codes on semaphore telegraphs, he wrote in desperation (in 1819):

> If an angel from heaven were to offer an unexceptionable plan, he would be ordered to walk, or fly, off with it, if he had not an article to offer which is not quite fit for angels to deal in, viz. Parliamentary interest. . . .
> Let it be recollected that a less simple plan, the establishment of mail coaches, was at the time deemed impractical and visionary. Man is a progressive animal; and it is the Press only that points out to him what he ought to do; and assuredly none but drivelling idiots, will oppose the march of knowledge. . .

How Best to Be Wrong

In rare cases even a potentially risky prediction about the potential of a new technology can prove to be a majestic underestimation. Witness this historic statement about the potential of electronic computers, made 45 years ago, in the March 1949 issue of *Popular Mechanics*:

> Where a calculator on the ENIAC is equipped with 18,000 vacuum tubes and weighs 30 tons, computers of the future may have only 1,000 vacuum tubes and perhaps weigh 1.5 tons.’

Wouldn’t it be nice if our guess about the potential of formal methods proves to be equally wrong?

REFERENCES

[BoHi94] J. P. Bowen, and M. G. Hinchey, *Seven more myths of formal methods*. Oxford University Computing Laboratory, Rep. PRG−TR−7−94, June 1994, 19 pgs

[Ch91] J. Chaves, *Formal methods at AT&T, an industrial usage report*. Proc. 4th FORTE Conference, Sydney, Australia, 1991, pp. 83–90.

[ClEm81] E. M. Clarke and E. A. Emerson, *Characterizing properties of parallel programs as fixpoints*. 7th Int. Coll. on Automata, Languages and Programming. LNCS 85, 1981.

[Dij75] E. W. Dijkstra, *Correctness concerns and, among other things, why they are resented*. Sigplan Notices, Vol. 10, No. 6, 1975, pp. 546−550.

[Dij76] E. W. Dijkstra, *A Discipline of Programming*. Prentice Hall, Englewood Cliffs, NJ, 1976.

[ELWW72] B. Elspas, K. N. Levitt, R. J. Waldinger, and A. Waksman, *An assessment of techniques for proving program correctness*. Computing Surveys, Vol. 4, No. 2, June 1972, pp. 97–147.

[Ha90] J. A. Hall, *Seven myths of formal methods*. IEEE Software, 7(5):11, 19, Sept. 1990.

[Ho81] G. J. Holzmann, *A theory for protocol validation*. Proc. 1st IFIP/INWG Symp. on Protocol Specification, Testing, and Verification. NPL, Teddington, England, 1981. Reprinted in IEEE Trans. on Computers, Vol C−31, No. 8, pp. 730–738.

[HoPa89] G. J. Holzmann and J. Patti, *Validating SDL specifications: an experiment*. Proc. 9th IFIP WG 6.1 Int. Workshop on Protocol Specification, Testing, and Verification, Twente, Netherlands. North−Holland Publ., Amsterdam, 1989.

[Ho91] G. J. Holzmann, *Design and validation of computer protocols*. Prentice Hall, Englewood Cliffs, NJ, 1991.

[Ho94] G. J. Holzmann, *The theory and practice of a formal method: NewCoRe*. Proc. 13th IFIP World Computer Congress, August 2 − September 2 1994, Hamburg, Germany.

[HoPel94] G. J. Holzmann and D. Peled, *An Improvement in Formal Verification*. This conference: Proc. 7th FORTE Conference, Bern, Sw., 1994.

[HoPehr95] G. J. Holzmann and B. Pehrson, *The Early History of Data Networks*. IEEE Computer Society Press, Los Alamitos, California, 1995, ISBN 0−8186−6782−6.

[Hou81] D. Hounshell, *Two paths to the telephone*. Scientific American, Jan. 1981, pp. 156–163.

[Mo61] P. and E. Morrison, *Charles Babbage and his calculating engines*. New York, Dover, 1961.

[Qu82] J. P. Queille, *Le système César: description, spécification et analyse des applications réparties*. Ph.D. Thesis, June 1982, Computer Science Dept., University of Grenoble, France.

[RuWe82] J. Rubin, J. and C. H. West, *An improved protocol validation technique*. Computer Networks, Vol. 6, Nr. 2, 1982, pp. 65−74.

[Ru92] H. Rudin, *Protocol development success stories: part I*. Proc. IFIP/PSTV Conference, Orlando, Fl., North−Holland, 1992, pp. 149–160.

[Su75] C. A. Sunshine, *Interprocess Communication Protocols for Computer Networks*. Ph.D. thesis 1975, Dept. of Computer Science, Stanford Univ., Stanford, CA.

[We93] C. H. West, *The challenges facing formal description techniques*. Proc. 6th FORTE Conference, Boston, Mass., 1993.