Predicting the Past

Gerard J. Holzmann

Shortly after the end of the Second World War there was a strong interest in improving the accuracy of weather forecasts. The importance of this was clear. If you can accurately predict storms, or any type of adverse weather, you have a much better chance to prepare for its effects and stay safe. George Dyson wrote about this in *Turing's Cathedral* [1] "In 1945 meteorology had become a science, while forecasting remained an art. […] On average, forecasts beyond twenty-four hours were still no better than 'persistence' – predicting that the weather tomorrow will be the same as it was today."

It was long realized that by dividing a larger area into a grid of small cells, with weather data like wind-speed, pressure, and temperature available for each cell, one could iteratively compute changing weather patterns over the entire area. Each cell would step by step only take the information from its neighboring cells into account to compute how the parameters gradually change over time. For sufficient detail, the cell sizes would have to be relatively small, which means that the amount of required computation would be massive: far exceeding what could be done with pen and paper by large numbers of human computers at the time.

Dyson quotes from a book written in 1922 by Lewis Richardson, who had been studying the potential of this type of method. The book titled *Weather Prediction by Numerical Process* [2], predicted "perhaps some day in the dim future, it will be possible to advance the computations faster than the weather advances, and at a cost less than the savings to mankind due to the information gained." At the time he estimated that about 64,000 human computers would be needed to make reasonably accurate weather predictions in this way across the globe. Although Richardson called that "a staggering figure," he probably had no idea just how far off the mark he still was.

The potential benefit of improved accuracy in weather predictions were not lost on John von Neumann either. In 1945, he used it as one of the selling points for his design of a new computer: the IAS machine. Von Neumann worked at the time at the Institute for Advanced Studies in Princeton (explaining the acronym IAS), where most work was focused on serious theoretical studies, so designing and building the hardware for an actual machine probably needed some strong justification.

Von Neumann recognized the need for more accurate weather predictions, and realized that he could use it to motivate an initially perhaps somewhat skeptical audience that work on the development of computing machinery could be worth it, in due time. A sobering thought, perhaps, is that today, with the availability of massively more weather data, and virtually unlimited amounts of compute power compared to von Neumann's days, we can still feel that the accuracy of weather prediction is often lacking. If the forecast says that it will be sunny for the next two weeks, few of us would be surprised if it still rained a few days later. Similarly, if the path of a hurricane is predicted to stay clear of your city one-week out, it may still be wise to prepare for a hurried evacuation later, just in case. We can all agree that the task is both very important and very difficult. The trouble in this case is that there is inevitably an element of randomness in weather patterns, which is sometimes jokingly referred to as the "butterfly effect." The cumulative effects of numerous small amounts of randomness can make it very hard to make precise long-term predictions, no matter how fast our machines are or how much data on current and historical weather patterns are available.

Predicting Software Failures

There's another type of problem that is of similar importance, and that is to "predict" more accurately how software or software systems may fail. Also in this case, especially in the last few years, through sites like GitHub etc., we have gained access to massive amounts of data on source code, and through databases like the Common Weakness Enumeration website (https://cwe.mitre.org) we now also have access to large repositories of software defects with documented security implications. Should all this not allow us to build better and better predictors for flagging vulnerable code? Also in this case the answer is mixed, but for somewhat different reasons than for accurate weather forecasting. Yes, we can scan code efficiently for the presence for any number of known vulnerabilities, and by doing so we can prevent the same types of vulnerabilities from causing harm. But, no, we are far from the point where we can give an accurate metric of an application's safety. That is if we exclude the easy answer to the question: "can this code fail in unpredictable way?" or "is this system vulnerable to attack by a determined adversary?" which in both cases is of course: "with virtual certainty: yes."

With software and cyber-security vulnerabilities we do not just have the problem that human designers and software developers can and do make both small and large random mistakes in the applications they develop. There is an added obstacle that weather forecasters do not have to worry about: the presence of highly skilled and determined hackers around the world, sometimes directly sponsored by foreign governments, that make it their life's mission to overcome every obstacle put in their way to disrupt critical systems, or to retrieve insufficiently protected data from them. They can afford to spend the time to pick apart software systems and to find just those nasty little random mistakes that make systems vulnerable to a takeover.

In his somewhat alarming new book "Sandworm," Andy Greenberg [3] spells out the consequences of this phenomenon of cyber-hacking when it is performed on a massive scale, and directed at large organizations or even at entire countries. Once hackers gain access to the critical infrastructure of industrial control systems, and they've shown repeatedly that they can do so with apparent ease, the potential damage is no longer restricted to some stolen credit card information or social security numbers. The hackers can knock out power systems, paralyze the control systems of hospitals, banks, and election systems, and famously in 2016 a group named the *Shadow Brokers* even succeeded in breaking into the highly protected systems at the NSA to steal much of its secret arsenal of tools. Would they have any trouble breaking into your home computer?

More Reliable Systems

To return to our analogy with weather forecasting: it is important to know if the area where you live is in the path of an oncoming hurricane, even if the predictions cannot always be perfect. Any type of reasonable forecast is better than none. With cyber-attacks all we need to know is that this new type of hurricane is indeed coming and it is coming our way with near certainty. But this is also a good reminder that reliability is fundamentally a *systems* property and not a component property. It is always wise to assume that any software application can fail, or can be hacked, but that does not mean that the system of which the software is part is just as vulnerable. We must learn to design (more) reliable systems from unreliable components. Also in this case, even small improvements can be very meaningful.

The principles of building reliable systems from unreliable parts were learned long ago in the design of mechanical systems. A good example is the use of the centrifugal governor on steam engines, as popularized by James Watt. It is a small device that detects overpressure in the steam engine's boiler

and automatically opens a valve to release that pressure to prevent the boiler from exploding. Similarly, in optical systems design we can use the flaws of one type of glass to correct for those of another, and thus create a system of lenses that is better than any of its components. In electrical systems, for instance in the design of signal amplifiers, we can duplicate the circuit to reduce the noise and improve the signal quality. The noise will be the one thing that differs in the output of the two subsystems that process the same input. Therefore, if we add the two outputs together, the original signal doubles in strength, but the noise signals (being uncorrelated) do not.

A simple method to setup your work system in such a way that it is protected better against complete data-loss in a targeted attack by a malicious adversary is to make sure that you always keep at least one trusted backup of your critical data completely offline, disconnected from all other devices and networks. Put more simply: don't leave your backup disk conveniently connected to your computer.

How we can protect our key infrastructure, like power, telecommunications, financial, or even election systems from silent hostile takeovers is still an open question. All these systems rely on software to operate correctly; software that in most cases was not designed to be incorruptible. How can we apply the principles of reliable system design in especially these cases? If Greenberg [3] is right, we may not have a lot of time left to figure that out.

References

1. George Dyson, *Turing's Cathedral – the origins of the digital universe*, Vintage Books, New York, 2012.
2. Lewis Fry Richardson, *Weather Prediction by Numerical Process*, Cambridge University Press, 1922.
3. Andy Greenberg, S*andworm – a new era of cyberwar and the hunt for the Kremlin's most dangerous hackers*, Doubleday, New York, 2019.