EDITOR IN CHIEF JEFFREY VOAS
IEEE Fellow; j.voas@ieee.org

Thoughts on Dependability

Jeffrey Voas, IEEE Fellow

Dependability suffers from various misunderstandings as to what the term means. Here, we discuss a few of these difficulties.

This April 2024 special issue discusses the current state of the practice and state of the art in *dependable computing*. To me, dependable computing is a *timely* and *timeless* topic. Many thanks to the guest editors for pulling this issue together. Here, to set the stage, I offer a message as to why I remain intrigued by this topic after so many years.

To begin, I came from the software side of the dependable computing research community as a graduate student. However, I was aware that the software side borrowed heavily from the hardware side (even though the sides had distinct differences). Let's walk through a few of them.

Hardware can fatigue, decay, and wear out. Software can't wear out, although its environment can. Wear out, decay, and fatigue are functions of *time*. Software is *deterministic*. Software's *static* behavior is not a function of time; however, its *environment* may be. Software's *dynamic* behavior is almost certainly a function of time.

For these and other reasons, software reliability measurement has suffered from *believability* and *acceptance* issues. It is difficult to believe that measures for *physical* systems can somehow be automatically mapped (with precision) to *nonphysical* systems.

Further, hardware testing is different than software testing. Hardware/physical testing is limited by time and money. For example, you can only crash-test so many cars. However, software testing is slightly different. The main limits on software testing are not the number of tests (because of the extreme performance of today's computers) but instead 1) being able to argue that the test cases were reasonable to select and 2) knowing if a test output is correct.

And then there's an implausible goal of detecting and "fixing" all faults. Does that make sense for faults of very small size, meaning those faults that are likely

DISCLAIMER

The author is completely responsible for the content in this message. The opinions expressed here are his.

Digital Object Identifier 10.1109/MC.2024.3350359
Date of current version: 5 April 2024

undetectable without using large numbers of tests? In most cases, it does not, except for critical systems.[1] And during testing, don't forget that rarely selected inputs can lead to rarely observed failures, and that some of these failures could be catastrophic.

Also, don't overlook the fact that software can be maliciously tampered with, and this tampering may be more difficult to detect than it is for hardware. Isn't it strange to think that you are more likely to *produce* correct software than to be able to *determine* that you did? And who knows where it goes once artificial intelligence-based code generation becomes normalized as a regular software development utility.

There is another dependability problem of applying *subjective*, everyday English words, like reliable, dependable, trustworthy, safe, and secure, to measures that are *objective*. For example, can you really say that a system is 100% reliable or safe and believe it? Or do you need to add a confidence measure (basically a disclaimer), so, for instance, you are 50% confident that the probability of failure is 1%? (I've always believed that it is better to use terms like "safer" or "more secure" than "safe" and "secure.")

For example, people often say that they want safe and secure schools. But what they really want is a reasonably secure school that is still *usable*, and if they get that, then they *should* get a *safer* environment for learning. True? And people always want security, but what if the security is unreliable? You could have the most elaborate home security system, but if it is down most of the time, then you don't have security. Here, a lack of reliability means a lack of security. As another example, polygraph tests were claimed to be 60% reliable in the 1950s; today that number is supposedly near 93% due to new computer technology, but 93% is still not reliable enough to be used in courts. The point here is that these everyday English words and how they are used complicates using them in technical measurements, metrology, and problem solving.

There is another dependability problem of applying *subjective*, everyday English words, like reliable, dependable, trustworthy, safe, and secure, to measures that are *objective*.

Another dependability problem involves "What is good enough?" and "Who signs off that a system is 'good enough'"? That is, who is *licensed* or *certified* to make such assertions? And *who* licenses or certifies the person making the assertions?[2,3]

To recap, what do we have so far? Theories for physical systems applied to nonphysical systems, subjective everyday English words being used objectively, and suspicious assertions about the trustworthiness made by fallible humans.

So, what could go wrong? Well, just about everything until you put measured *boundaries* on assertions of dependability. Once you take this approach, you can begin to sound reasonable. For example, you can say that, given these X restrictions/assumptions, we can assert that this system cannot exhibit these Y behaviors, where X defines the usage environment and possibly other conditions. By doing so, we can argue that a system is *fit for purpose*. But that is still only an argument and not an iron-clad guarantee.

Systems have many dimensions to consider before we label or assert their dependability.[4] Dependability is a complicated recipe that is best served with a cup of salt.[5,6] C

REFERENCES

1. J. Voas and K. Miller, "Predicting software minimum-time-to-hazard and mean-time-to-hazard for rare input events," in *Proc. IEEE Int. Symp. Softw. Rel. Eng.*, Toulouse, France, Oct. 1995, pp. 229–238, doi: 10.1109/ISSRE.1995.497662.
2. K. Miller and J. Voas, "Software certification services: Encouraging trust and reasonable expectations," *IT Prof.*, vol. 8, no. 5, pp. 39–44, Sep./Oct. 2006, doi: 10.1109/MITP.2006.120.
3. J. Voas and P. Laplante, "The services paradigm: Who can you trust?" *IT Prof.*, vol. 9, no. 3, pp. 58–61, May/Jun. 2007, doi: 10.1109/MITP.2007.59.
4. K. Miller, J. Voas, and P. Laplante, "In trust we trust," *Computer*, vol. 43, no. 10, pp. 85–87, Oct. 2010, doi: 10.1109/MC.2010.289.
5. J. Voas, "Trusted software's holy grail," *Softw. Qual. J.*, vol. 11, no. 1, pp. 9–17, May 2003, doi: 10.1023/A:1023679926998.
6. J. Voas, "Software's secret sauce: The "-ilities" [Software Quality]," *IEEE Softw.*, vol. 21, no. 6, pp. 14–15, Nov./Dec. 2004, doi: 10.1109/MS.2004.54.

JEFFREY VOAS, Gaithersburg, MD 20899 USA, is the editor in chief of *Computer*. He is a Fellow of IEEE. Contact him at j.voas@ieee.org.