IoT Metrology

Jeffrey Voas
NIST

Rick Kuhn
NIST

Phillip A. Laplante
Penn State University

Metrics, measurement, and metrology are different but related concepts that are essential for creating standards for physical systems, virtual systems, financial institutions, medical care, first responders, governance, and others.[1,2]

We depend on standards for nearly everything. Standards are based on measurement—without measurement, standards are difficult to write, and determining compliance for a standard becomes far more difficult.

Metrics and measures are classified in many ways. We begin by looking at a few of these classifications.

Measurement has its own field in science: *metrology*. Measurement can be used in determining what we currently have and what we can expect in the future based on what we have. The first is generally easier to measure. For example, we can count the coffee beans in a bag to determine what we have. But knowing whether those beans will result in a good cup of coffee once the beans are ground is a different question, unless there are almost no beans in the bag.

Estimation and prediction naturally use numerical measures. Estimation tells you approximately what you have today with respect to a fixed environment and context. For example, a system might be estimated to be 99 percent reliable today. Note the key word is *today*, a point in time. In comparison, a prediction would say something like, "Based on an estimate of 99 percent reliability today, we believe it will also be 99 percent reliable tomorrow, but after tomorrow, the reliability might change." Why? As time moves forward, components might wear out, thus reducing overall system reliability. And as time moves forward, the environment might change so that the system is under less stress, thus increasing the predicted reliability.

We should also address the difference between *quantified measures* and *qualified metrics*. A quantified measure is numerical, like the example of 99 percent reliability. A qualified metric would be something crude like a red, green, yellow, or orange threat level, or a capability maturity model (CMM) level of 1 to 5. Metrics use measurement and other information to describe a product or process.

Metrics related to software code generally have primarily *syntactic* or *semantic* aspects. A syntactic measure would include such criteria as the number of conditional statements, loops, or variables in a block of code. A semantic measure could be, for example, the fact that half of the code is dead because no input exists that can reach that code upon execution. Note that in semantic measures, meaning is almost always a function of the operational environment and the context in which the code executes.

Metrics can also have static or dynamic characteristics. Syntactic measures are static; semantic measures are usually dynamic. Environment and context give semantics to static syntax. Environment and context provide the notion of "dynamic." As we mentioned with reliability, dynamic measures also include one other very important variable: time. Dynamic metrics are

applied to systems in operation—for example, a failure rate of a system over *x* units of time is a dynamic metric.

Finally, it is worth noting the difference between networked systems that are bounded versus unbounded. This is an issue because quantifying a system without some limit on the system's bounds is not plausible nor defensible. Numerical measures only make sense for systems in which bounds can be defined. Unbounded systems are far better served by qualifiable measures.

The point here is that the field of metrology has evolved into many classes and viewpoints over centuries. How might that body of knowledge be applicable to the Internet of Things (IoT)? Research opportunities for new measures that are IoT-centric might be in plain view but appear to be hiding. What might IoT metrology look like if we were to create this sub-field based on what we can leverage from the past?

What might IoT metrology look like if we were to create this sub-field based on what we can leverage from the past?

PREVIOUS COUNTING MEASURES

In IT system measurement, we have easily counted items such as nodes, clients, gateways, servers, and so on. While simply counting items is crude, it offers hints concerning size and complexity. And by counting gateways to the outside world, hints are offered into how security might be compromised. IT system measurement is a challenge. The benefit of adding up simple counts for the "pieces and parts" offers crude insights into complexity, maintainability, size, security, and testing costs.

In software measurement, we have similar count measures. We measure source lines of code (SLOC), branches (through the code), the number of test cases needed to test the branches, complexity, and other related or derived measures. (Paths are another measure that are sometimes of interest, but are usually dynamic because the number of paths is usually infinite and therefore not countable.) IT system measurement and software measurement have a history of similar counting approaches.

INTERNET OF THINGS

There is almost no published work on metrics, measurement, or metrology for the IoT. One notable exception is Paschou et al., who largely focused on data throughput measurement in healthcare applications.[3] Perhaps the dearth of research is based on the following question: In the IoT, what can we measure? IoT is an acronym of three letters—the "I" (Internet) existed long before the acronym was termed, so "T" (things) is the letter in the acronym that matters. To assist with this understanding, NIST mapped out five Network of Things (NoT) primitives.[4]

NoT is a term that applies to both cyber-physical systems and the IoT. The five NIST primitives of all NoT systems include:

- sensor (something that measures physical properties),
- aggregator (software to transform data from a sensor),
- communication channel (data transmission, such as wired or wireless),
- e-Utility (software or hardware to execute processes, such as a database), and
- decision trigger (produces the final result, such as an output signal to an actuator).

These five primitives define the "Lego-like" building blocks used by any IoT-based system. The primitives are the Ts. The easiest way to think about this is that the "things" are what make up the IoT. That might offer a partial hint into IoT metrology. So, we will explore what metrics and

measures we can offer related to these five classes of "things" as well as to their interactions given an IoT architecture (system) operating in real time.

COUNTING: "LOW-HANGING FRUIT" APPROACH

Some question whether the IoT is simply marketing hype or whether there is a science behind it. That's a fair question. So, what is the IoT? After all, we should answer that before we start measuring. However, we have no intention of defining the IoT here. That is a different discussion. Fortunately, and instead, we can employ the five "Lego-like" building blocks to seek metrics that are collectable.

The first four primitives are easily countable, and it is likely that the fifth primitive will be of size one as there often will only be one decision trigger. Counting "things" correlates nicely with previous measures such as SLOC, nodes, clients, gateways, and servers.

So, what does counting "things" offer? It offers a crude static view of complexity. It is a "pieces and parts" approach. Counting is no different than the coffee beans in the bag—it is boundable. The bag can only hold so many beans. But unlike coffee beans, nodes in a network interact, and potential interactions between pairs of nodes rise and fall quadratically as the count of nodes increases or decreases. So, a count of "things" offers useful, though still crude, bounds on complexity that can be refined by looking at the connections among "things."

INTERCONNECTIONS: "MIDDLE-HANGING FRUIT"

Measurement of a system is different than the measurement of a smaller unit such as a "thing."[5] For example, if A is measured for metric X and B is measured for metric X, and somehow A and B are composed, what is the resulting new value of metric X? If X is a measure like SLOC, it's simply additive counting. It will be more problematic if assessing the semantics of a newly composed "thing." For example, if X is reliability, counting will not work. Reliability metric X was assessed when A and B were standalone "things." After composition, they are no longer standalone. You cannot add their reliabilities together. However, results from network queueing theory could be applied to begin to quantify NoT reliability. One of the challenges for IoT engineering will be adapting queueing theory approaches developed for largely static networks to the typically larger (by node count) and much more dynamic NoTs.

As the five classes of "things" interconnect, larger "things" are built. You can think of these as sub-NoTs or subcomponents—in other words, larger "things" made up of smaller ones. These subcomponents could be as simple as one sensor connected to its corresponding communication channel. While this sounds trivial, those two "things" will need to interoperate correctly and at the appropriate time. You can think of these composite "things" as what was termed "programming patterns" years ago.

With a two-"thing" component such as a sensor connected to a communication channel, you can test it and assess a semantic measure such as performance. For example, does the sensor provide its data quickly enough, and when it transfers its data to a communication channel, does the communication channel move the data quickly enough to the next "thing"? In this case, you would have a dynamic measure (performance) of an IoT pattern given a fixed test distribution.

TRUST MEASURES

At best, any definition of trust with respect to the IoT should be viewed suspiciously because we are without an accepted and actionable definition of the IoT. A reasonable approach to measure trust is probably to start with existing well-defined dynamic metrics, commonly referred to as the "-ilities."[6,7] Examples include reliability, maintainability, testability, security, safety, privacy, performance sustainability, fault tolerance, and resilience. Some of these are quantifiable, like reliability, but most are not. For example, the general definition of "reliability" is the probability

of failure-free operation of components or a system for a fixed interval of time and in a fixed environment. Note the importance of the term "fixed." The term "fixed" here is essentially the same as the term "bounded," which we already argued is necessary in numerical measurements.

In contrast, security is generally unbounded. Security suffers from an unknown and therefore unbounded threat space. The set of exploits (generally unknown) to take advantage of the set of vulnerabilities (generally unknown) creates this bounding problem.

There might be some "-ilities" that are best served by regulated or legislated policies. While policies are not metrics, they are quasi-standards and rules for how processes and systems are expected to operate. Note, however, that policies without measures to determine compliance are easy to circumvent.

It has often been written that an "-ility" such as security is at odds with another, such as performance. Consider the example of airport security, where the more security tests that each passenger must go through, the slower the line of people moves. There are other examples of "-ilities" in conflict. Consider blockchain. One of the claimed benefits of blockchain is increased trust due to transparency. However, privacy is also usually considered an attribute of trust. Transparency and privacy are at odds. So, tradeoffs among such conflicting "-ilities" need to be considered when creating a qualified security measure such as red, green, yellow, and so on. This can be done by using weighting factors of importance that are applied to each individual measure.

> A reasonable approach to measure trust is to start with existing well-defined dynamic metrics, commonly referred to as the "-ilities."

DASHBOARDS

Software measurement dashboards were popular in the 1990s and early 2000s, and were usually used for project management reporting. The general idea was to roll up low-level measures into visual reports so it was easy for program managers to see where a project was in terms of plan and schedule. Might such dashboards be useful for NoTs in operation? Unfortunately, this is not practical yet for the IoT, but we contend that with internal probes—sometimes called "assertions"—dashboards might be a reasonable approach for measuring an NoT during operational usage.[8] Further, dashboards can be leveraged to create qualified measures such as red, green, and yellow. This might be a first attempt at creating system-wide dynamic IoT measures.

CONCLUSION

There are no simple metrics for IoT systems that are highly dynamic and operating in real time. To date, there are no dashboards, rules for interoperability and composability, rules of trust, established approaches to testing,[8] or even an accepted and actionable definition of the IoT.

Our goal is not to discourage research into the developing new metrics that are aligned to this new distributed computing technology or to discourage attempts to apply past metrics to the IoT. We acknowledge that there are difficulties, but we are optimistic because we have a rich history of previous measurement approaches that we can leverage.

Finally, we did not consider financial metrics or metrics for user satisfaction, convenience, safety, and other important qualities. These will be important in judging whether a purposed NoT is worthwhile, and metrics for these qualities will likely depend on a body of scientific measures of IoT characteristics, such as those we have discussed.

Note: *Identification of products does not imply endorsement by NIST, nor that products identified are necessarily the best available for the purpose.*

REFERENCES

1. J. Voas and R. Kuhn, “What Happened to Software Metrics?,” *Computer*, vol. 50, no. 5, 2017, pp. 88–98.
2. P.A. Laplante, J. Voas, and N. Laplante, “Standards for the Internet of Things: A Case Study in Disaster Response,” *Computer*, vol. 49, no. 5, 2016, pp. 87–90.
3. M. Paschou et al., “Health Internet of Things: Metrics and Methods for Efficient Data Transfer,” *Simulation Modelling Practice and Theory*, vol. 34, 2013, pp. 186–199.
4. J. Voas, *Networks of ‘Things'*, government report NIST SP 800-183, NIST, 2016; doi.org/10.6028/NIST.SP.800-183.
5. J. Voas and P.A. Laplante, “IoT’s Certification Quagmire,” *Computer*, vol. 51, no. 4, 2018, pp. 86–89.
6. J. Voas and G. Hurlburt, “Third Party Software’s Trust Quagmire,” *Computer*, vol. 48, no. 12, 2015, pp. 80–87.
7. J. Voas, “Software’s Secret Sauce: The "-ilities",” *IEEE Software*, vol. 21, no. 6, 2004, pp. 14–15.
8. J. Voas, R. Kuhn, and P.A. Laplante, “Testing IoT-Based Systems,” *12th Int'l Symp. Service-Oriented System Engineering* (SOSE), 2018.

ABOUT THE AUTHORS

Jeffrey Voas is a computer scientist at NIST. Contact him at jeff.voas@nist.gov.

Rick Kuhn is a computer scientist at NIST. Contact him at kuhn@nist.gov.

Phillip A. Laplante is a professor of software and systems engineering at Penn State University. He is an IEEE Fellow. Contact him at plaplante@psu.edu.