From the Editors

No Preconceptions

Jeffrey Voas
Fellow, IEEE

I AM GENERALLY a nonreader. As far back as elementary school, I found the demands to read annoying. I was the kid looking for Cliffs notes or asking others to tell me what the book said before I went into a class so that I could pretend I had read it.

College was no different. I found textbooks boring and poorly written. So, I adapted. I found an approach that probably does not work for many people, but it did work for me: I look at a problem not knowing what others have written about it and see if I could solve it on my own. For subjects like calculus, no luck. But for my Ph.D. dissertation, it worked fine. Starting from nothing, I had a dissertation written and defended in 20 months.

My dissertation topic started with a simple goal: Create a reliability model for white-box software (software whose source code can be examined). My advisor gave the seminal book at that time on software reliability, but it was focused on reliability models for black-box software (software whose source code is not available and often proprietary). I read a few chapters, gave up on the book, and decided to start anew. I started with no preconceptions—a blank slate.

I began from the three necessary and sufficient conditions needed for software to fail. They are execute a fault (commonly referred to as a "bug"; mistakes in the code are as unavoidable as death and taxes); then have that fault infect the internal state of computation; and then propagate that infection to create a failed output. This perspective provided the white-box flavor I needed, and that allowed me to look at how those conditions affect probabilities of failure. (And the relationship between a probability of failure and a reliability estimate is simple: essentially one minus the probability of failure equals the reliability.) Every reliability, safety, fault tolerance, and testing approach used today is in some way based on these three conditions occurring. My white-box approach estimated the probability of those three conditions occurring. That was new, and it was good enough to graduate.

That underlying model with the three necessary and sufficient conditions has never let me down. It allowed me to create new ways to inject faults into software to study how the software reacts—similar to "fuzzing," where you try to break a system in order to figure out how to strengthen it—back in the 1990s. It allowed me to create new ways to test and prove that software is working—called certification—in the early 2000s.

After rolling-off a research effort on vetting mobile apps, I was interested in researching Internet of things (IoT) security. I had no idea what IoT was except from what I had heard at a few keynote talks, and I assumed that since there was much talk about IoT in the public space, there surely must be a "simple to grab and use" definition from which I could start. I was wrong.

Digital Object Identifier 10.1109/MITP.2019.2897262
Date of current version 27 March 2019.

Published by the IEEE Computer Society

Worse, there was not even a "close to usable" or modifiable definition.

So, like in my past, I relied on trying to "roll my own" definition.

I began from what I had heard: Many sensors would be involved and there would be much data, and that data would somehow drive decision-making, automation, and analytics. (Some talks were so over the top they described it as the last great technology the world would ever need.) It was also obvious that for all of this to work in unison, there had to be ways for data to move from one "thing" to another, although what a "thing" was still needed serious contemplation. It was also obvious that massive amounts of data would somehow need to be reduced to lesser amounts of information to drive decision-making. And to reduce information and perform decision-making, computational engines would be needed. So, all of this taken together offered me a foggy sense of what IoT might be about.

This simplistic idea eventually culminated in an idea I coined called "network of things." The idea discussed five basic Lego-like parts: sensors, aggregators, communication channels, external utilities, and decision triggers. Sensors are self-explanatory, aggregators are the tools to reduce much information to less information, communication channels allow "things" to talk to "things" and provide all data flow, external utilities execute all computations, and decision triggers provide decision-making. These simple categories of "things" are the building blocks for any network of "things," including a smart car, smart home, smart TV, etc.

But only spelling out these five types of "things" still did not get me to a definition of IoT. Was that a mistake or intentional? It was neither. It was simply my good luck.

So, why do I say that? It goes back to when the term IoT was first published. The term was coined by Kevin Ashton in 1999, although he apparently later preferred the phrase "Internet for things." Of the three letters in IoT, the (o) matters little, and as already mentioned, "of" might be better replaced by "for." The Internet (I) existed long before the IoT acronym and term was coined, and so it is the "things" (T) that makes IoT subtly different from previous IT systems and computing architectures. The "things" are IoT's secret sauce.

By postulating that a network of "things" model might be more usable and actionable than an IoT model, I avoided having to define what the Internet is and what all of the specific "things" might someday be. That is useful because the state and extent of the Internet changes so quickly that it cannot be described, or bounded, for much longer than an instant, so it is effectively not boundable. And it is very hard to define something that you cannot bound, like the universe.

So, by not defining IoT, I gained the luxury of not answering questions like "do you have an IoT? And if so, can you show it to me?" Instead, since a network of "things" can be bounded, I can now query about your network of "things" by asking questions like How many 'things' are in your network, and how many of them are sensors?" or "Do you use public clouds to handle your computations, and if so, how many different heterogeneous clouds do you communicate with?," or "Do you have performance issues with the latency in your communication channels?" The point here is that the concept of a network of "things" frees us to compare, measure, and contrast basically any bounded network. So now, there is an answer for people who question whether IoT is just marketing hype or if there is science and engineering behind it. There is science and engineering behind it!

So, what am I saying? By dispensing with common wisdom, and in this case, the existing belief that we must have a definition for IoT, I opted for an engineering-based view describing the building blocks of IoT. It came from one simple change in mindset, made possible by having no preconceptions.

Disclaimer

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