Some computer scientists and software engineers write about the history of their fields. Others analyze and document the philosophy of their own discipline. In this latter regard, I am happy to announce the publication of Edward A. Lee's book on the philosophy of engineering:


Lee's research at Berkeley centers on the role of models in the engineering of cyber-physical systems. That's what he has in common with Hermann Kopetz, Michael Jackson, David Parnas, and many others.

I have only ordered and not yet been able to read Lee's book. Lee and I have exchanged intellectual thoughts during the past few months, mostly in connection with the Kopetz Principle (see below). In the present post — Part 1 — I will look forward and speculate on likely topics in Lee's book. In a subsequent post — Part 2 — I will dive into specific sections of the book and discuss some technical topics from my vantage point.

For those interested in original views on the philosophy of computation, rest assured that Lee is not one of many who promotes the idea of digital physics. The physical world is not equivalent to software, according to Lee. The “likelihood is remote,” he writes on the book's website, “that nature has limited itself to only processes that conform with today’s notion of digital computation.”

For those few — such as Peter Naur, Michael Jackson, and Dave Parnas — who are well aware that computer scientists typically mistake their favorite model for reality, rest assured. For, that message is precisely one of Lee's central points in his recent publications, talks, and in his 2017 book. Models are “invented not discovered,” Lee conveys on the book's website, “and it is the usefulness of models, not their truth, that gives them value.” This insight is much in line with Naur's views [1]. Lee also writes that models “are best viewed as having a separate reality from the physical world, despite existing in the physical world.” This, too, is not unlike the ideas put forth by Brian Cantwell Smith [11], James Fetzer [5], and especially by Jackson [7]. One aspect that definitely sets Lee apart from many of the other aforementioned scholars is his extensive expertise in physics and engineering. So, at the very least I expect Lee's 2017 book to be refreshing. I anticipate encountering arguments that complement those provided in the secondary literature, if not shed new light with unmatched technical precision. (For example, Lee’s research on determinism, concurrency, and on timed systems is definitely new to me. And I personally look forward to examining how Lee reasons about undecidability in Chapter 8 of his book.)

Finally, for those of us who are highly skeptical about the Internet of Moving Things, it might be worth our while to examine Lee's seemingly optimistic account in this regard. On the book's website, Lee argues that

> the pace of technological progress in our current culture is more limited by our human inability to assimilate new paradigms than by any physical limitations of the technology.

I am prepared to scrutinize Lee's thoughts on this matter because I personally think that there are systems that are simply not makeable. (I am currently not in a position to thoroughly judge whether Lee would agree or disagree with my own take on this topic.) For example, today in Belgian newspapers: thousands of deployed pacemakers need to be upgraded as soon as possible due to a security vulnerability. That really is a déjà vu for me. (I have raised similar concerns pertaining to automated cars, drones, smart phones, and nuke subs.) If I were to carry a pacemaker, I would prefer it to be one that is not connected to a network, even if that would mean that I would have to pay more for such allegedly outdated technology. Software realists know that the next generation of pacemakers will still contain security vulnerabilities; it is only a matter of time before someone
discovers them. Surely, there must be more people who share my frustration with this techno hype. (Indeed, fortunately there are policy makers who do listen to techno skeptics. I am referring to the Dutch elections held earlier this year, as reported briefly here.)

So, I am eager to find out what Lee’s stance is on the viability of having a humanly safe Internet of Moving Things. Specifically, does Lee address, directly or indirectly, the following fundamental question?

\[
\text{Will we (ever) be able to know, let alone convincingly demonstrate to others, that our software is bug free?}
\]

If we will never achieve this goal, then neither can we guarantee that our software is completely secure. For, even a bug-free C program can be vulnerable to a malicious hacker [10, Sec.IV]. In my own words:

**Bug-free software is a necessary, but not a sufficient, condition for having a secure—and, hence, a humanly safe—cyber-physical system.**

At the end of the day it is all about human safety, not correctness, nor security. I don’t mind having bugs in a text editor or an unmanned Mars Lander. But I will mind the bugs in fast moving objects (such as drones and automated vehicles) that society at large will use on a daily basis, because those bugs can have a detrimental effect on human safety.

There are laws for software engineering that have yet to be widely disseminated and I consider the previous bold-faced passage to convey one such law. I want to find out whether Lee shares my inclination and realism, because being a lonely scholar only gets me so far.

In the rest of the present post I will focus on the role of models, as perceived by Lee and like-minded researchers and starting with Naur.

**PETER NAUR**

In 2011 I interviewed Peter Naur at his home in Denmark. Naur’s criticism for formal methods was two-fold:

1. His plea for pluralism: a computer professional should not dogmatically advocate a formal method and require others to use it in their own work.
2. Naur’s insistence to distinguish between models on the one hand and the things being modeled on the other hand:

   A model of the structure of a molecule is just one kind of description of the substance under investigation, well known to ignore certain aspects of the actual substance .... [1, p.86]

   Descriptions are not true, never, they are more or less adequate. That’s the best they can offer. [1, p.86]

Continuing on my quest to understand the history of formal methods in computer science, I also interviewed Michael Jackson.

**MICHAEL JACKSON**

It is Jackson’s reversal of Dijkstra’s famous sentence, which I find to be most rewarding:

\[
\text{[P]rogram testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence.}
\]

— Dijkstra [3].

To turn Dijkstra’s famous observation on its head, we should say that in a cyber-physical system, where the computer’s purpose is to exercise control over the material world, formal reasoning can demonstrate the presence of error, but never its absence.

— Jackson [7, p.67].
Jackson has been inspired by Cantwell Smith’s famous paper, ‘The Limits of Correctness’ [11], and it should also be noted for the sake of completeness, that Jackson’s original words appeared in 1998:

Dijkstra famously observed that in the pursuit of program correctness “testing can show the presence of errors but not their absence”. In the pursuit of well-engineered descriptions of the real world we should recognise — and every student of software engineering should be taught — that formalisation can show the presence of description errors, but not their absence. [6, my emphasis]

HERMANN KOPETZ

More recently, and thanks to a pointer provided by Jackson, I came across the writings and YouTube videos of Edward A. Lee. Specifically, Lee conveyed the revealing “Kopetz Principle” in his 2012 talk, entitled: ‘Verifying Real-Time Software is Not Reasonable (Today).’ Indeed, 15 minutes into his talk, Lee provides the following text:

The Kopetz Principle
Many (predictive) properties that we assert about systems (determinism, timeliness, reliability) are in fact not properties of an implemented system, but rather properties of a model of the system.

We can make definitive statements about models, from which we can infer properties of system realizations. The validity of this inference depends on model fidelity, which is always approximate.

This passage is a result of Lee paraphrasing another expert in real-time software: Hermann Kopetz. Some 45 seconds later, Lee comments on formal verification in particular:

The proofs in verification are not proofs about the system but about a model of the system. The usefulness of these proofs depends on the model fidelity. The model is always an approximation. [This is my paraphrased version of Lee’s oral comments.]

Finally, and unsurprisingly, Lee also disseminates these insights via his research papers. For example, he insists that a C computer program is a model too and not a system realization [8, p.4839]. The question I have, and which I will keep in the back of my mind while reading Lee’s 2017 book, is whether that model is a mathematical model and, if so, what type of mathematical model. (I, myself, take a C computer program to be a non-mathematical model and, specifically, a technical artefact [2], following Raymond Turner [12]. That artefact can, of course, be mathematically modeled in various ways, resulting in multiple mathematical models of the C computer program at hand.)

DAVID PARNAS

Mistaking the model for reality is also what Parnas is concerned about [2, p.8]. Rather than verify properties of an actual program, computer scientists are examining models of programs [2, p.100]. That, again, is very much in line with the Kopetz Principle. Moreover, last week I came across Parnas’s article ‘The use of mathematics in software quality assurance’ [9]. Especially page 14 is to the point. Parnas warns his readers to use a model with great care: “the user must always be aware that information obtained by analyzing the model might not apply to the actual product.” Likewise:

Most of the many published proofs of programs are actually proofs about models of programs, models that ignore the very properties of digital computers that cause many of the ‘bugs’ we are trying to eliminate. [9]

Again, these ideas can already be found to some extent in the writings of Cantwell Smith [11], Fetzer [4,5], and others, but Parnas is zooming in on the Formal Methods community and is writing about formal verification. That’s why his writings, now adjoined with Lee’s 2017 book, will probably remain on my desk for the years to come.

REFERENCES
writing and speaking sections of the exam. We hope that you find the modules in this book useful and that the vocabulary you acquire will help you to achieve the grade you want in the IELT Peter Drucker (1909-2005) was one of the most widely-known and influential thinkers on management, whose work continues to be used by managers worldwide. He was a prolific author, and among the first (after Taylor and Fayol) to depict management as a distinct function and being a manager as a distinct responsibility. Throughout his long career he has had interests as diverse as journalism, art appreciation, mountaineering, reading - drawing inspiration from the works of Jane Austen - and, of course, management teaching, writing and consultancy. With 39 books published over seven decades (and translated into at least 30 languages) and many books written about him since his death, Drucker was, by common consent, the founding father of modern management studies. Biography.