Editor: **Philippe Kruchten**
University of British Columbia
pbk@ece.ubc.ca

A Manifesto for Energy-Aware Software

Alcides Fonseca, Rick Kazman, and Patricia Lago

ACCORDING TO RECENT estimates, computing and communications could account for 20% of energy usage globally by 2025.[1] This trend shows no sign of slowing. The annual growth in power consumption of Internet-connected devices is 20%. Data centers alone are now accounting for more than 3% of global emissions. Even if you are not worried about this trend on the mega scale, you are likely concerned with the power consumption of the devices in your pocket, on your wrist, and in your ears.

Software, hardware, and network attributes all contribute to power usage, but little attention has been given to this topic by the information and communications technology (ICT) community. For example, as software engineers, we were never taught to consider, much less manage, the energy consumption of the software systems we created. Despite our lack of awareness and preparation, we are now facing an undeniable reality: the software community must learn to design for, monitor, and manage the energy usage of software. For this reason, we argue the need for energy-aware software and present a manifesto describing nine guiding principles. By *energy-aware software*, we mean software that is consciously designed and developed to monitor and react to energy preferences and usage. Energy efficiency is, therefore, one possible (but not the only) response to being energy aware.

This manifesto and the principles it proposes have arisen from our experience and from the experience of more than 100 researchers and practitioners who have participated in six international workshops on the engineering of green and sustainable software.[2] Why do we need a manifesto? Why now? Although there has been some attention to this area,[3] we believe it has been grossly insufficient given the high stakes involved. The vast majority of practitioners (and researchers) are completely ignorant of energy concerns; they, and the programs they create, are anything but energy aware.[4]

The Nine Principles of Energy Awareness

Energy awareness is a necessary but not sufficient precondition for energy efficiency. Energy awareness is required from all stakeholders, such as end users who may choose product A versus product B based on energy characteristics. Our goal in this manifesto is to call for changes in how we think and what we do. This will not come for free, but we believe that the cost of inaction is far greater.

Public Awareness Is Key for Widespread Adoption

We believe that the key to widespread adoption of energy-aware software is to sensitize and empower end users. The scary statistics regularly published have proven ineffective so far (the amount of energy being consumed by ICT, the increasing amount of energy consumed by cloud providers, the massive amounts of data being stored in data centers as opposed to the negligible percentages of data being actually used, and so forth). Neither do the worrisome energy-consumption predictions seem to spur us to action (such as the increasing number of things being connected to the Internet or the booming growth in mobile devices and their increasingly sophisticated applications).

We need to turn these alarming trends into an opportunity: 1) to sensitize end users to the amount of energy consumed by the software they use and 2) to create awareness of

Digital Object Identifier 10.1109/MS.2019.2924498
Date of current version: 22 October 2019

the fact that software solutions with similar features may yield very different energy profiles. Imagine that we were able to attach "green" labels (like Energy Star ratings) to the apps available in Google Play or the Apple Store. End users of mobile devices could then compare the apps they are seeking (such as apps for "meditation" or for "scanning documents") not only in terms of features, rating, and price; if green labels were added, end users would be empowered to make better-informed decisions based on the labeled level of energy use. Such labels would force software companies to invest in optimizing the energy impact of their applications (if they wanted to improve their market position), with a resulting positive effect on the resources (for example, cloud services and networks) that such applications use.

Incentives for Software Stakeholders Should Be Provided

We believe that, although some people are altruistic, most people respond to incentives. Therefore, such incentives should be put in place to encourage the creation of energy-aware systems, which will, in turn, lead to energy efficiency. Such incentives would also help raise the consciousness of engineers and end users. Most stakeholders need to be incentivized to actively pursue software and systems that are energy aware. This is a key step to raising the priority of energy-aware software in our companies and in our society. Such incentives, both positive and negative, have been used successfully for decades in other parts of our economy: there are so-called sin taxes to discourage drinking and smoking, and there are tax rebates to encourage people to make their homes more energy efficient.

Energy-Aware Software Engineering Should Be a Priority for Every Stakeholder

The efficiency of system energy consumption is relevant for everyone, regardless of their roles in the development process—from end users, concerned with their battery life, to business owners, concerned with reducing their electricity costs and from the developers, who should understand the energy impact of their contributions, to the product owners, who have to decide in what way energy efficiency is a requirement.

Energy-related information should flow to all stakeholders. Early in the development process, clients should be asked about their energy requirements. These should then be propagated to developers, testers, and operations personnel so that they can be taken into account, tested for, and monitored. Results from these phases should be reported back to the stakeholders, just like other important metrics.

Education and Professional Training Should Cover Energy-Aware Software by Default

To create energy-aware software, we must educate the next generation of engineers who need to acquire the competencies (and provide training for the current-generation workforce, too). Depending on the target audience and the learning objectives, educational programs may adopt a centralized approach (concentrating the competencies crucial for energy-aware systems in one or two courses), distributed (revising traditional courses to include competencies for energy awareness), or blended (including both types of courses across the curriculum). Just as every programmer should understand algorithmic complexity, energy awareness must become a standard competency of every ICT practitioner and a standard consideration for every decision maker and end user.

Broad Adoption Requires Attention to Usability

To encourage broad adoption of energy-aware systems, we must pay attention to the usability, from a developer perspective, of the tools that we create. Best practices for energy-aware software engineering should end up being embedded in the tools, packages, and frameworks we create so that software engineers do not need to reinvent the wheel. You should not need a soldering iron or a circuit diagram—or a degree in electrical engineering—to manage the energy consumption of an ICT system. Furthermore, it should be easy to reuse experience and best practices for engineering energy-aware software.

Energy Awareness Should Be Engineered Throughout the Lifecycle

Energy awareness can and should be treated like an architectural quality attribute,[5] no different from how we design for, analyze, prototype, and manage other qualities in an architecture such as modifiability, performance, availability, or security. This means that architectures are design blueprints with system-wide resource-management strategies. For example, power usage requirements should be explicitly collected during requirements gathering, designed for, and tested for.

In particular, energy awareness and energy efficiency must be designed into a system early in its lifecycle and considered when making major changes to the system. Leaving this consideration until the system is already built is a recipe for

disaster. Experience tells that any quality that you address late in development tends to be treated superficially (or at great cost). If you don't measure it, you cannot manage it.[6] This energy awareness will have an associated cost, in terms of system complexity, and this cost must be acknowledged and accepted by the ICT community.

Software Quality Should Not Come at the Expense of Energy Awareness

Energy-aware software development does not imply that energy efficiency should be prioritized over other quality attributes. Being energy aware involves taking into account the energy consumption of software across the software development lifecycle. Energy awareness cannot be ignored and should be explicitly considered in tradeoff decisions, even if the final decision is to prioritize some other quality attribute over it.

Making software development processes explicitly energy aware allows for stakeholders to be informed of the options chosen regarding the energy consumption of software. Data about the energy impact of design choices can be used to inform future decisions and improve designs with respect to their energy efficiency.

Energy Awareness Demands Dynamic Adaptability

Energy awareness is heavily influenced by the context in which software is being used, both because resource availability varies over time (for example, the battery load is limited, network connectivity is location dependent, or electricity rates change during the day) and because as end users move, their needs change (for example, driving the most energy-aware route depends on our location, and if we need to charge our car battery, it depends on the availability of charging stations nearby).

We believe that software should perform its core functionality while simultaneously ensuring energy awareness. If this happens, end users can count on software applications to be reliable and to promise the best tradeoff between energy and functionality. To do so, software must be able to detect that its context has changed and that (possibly) some resources have become scarce and flexibly adapt by replacing them with alternatives or downgrading the delivered functionality.

We Value Measures Over Beliefs (and Reliable Trends Over Precision)

Energy awareness can be easily ignored early in the software lifecycle if there are no data to support the fact that energy should be a concern. Another common pitfall is to believe that optimizing for energy efficiency is difficult, if not impossible.[7] Measuring energy consumption is a first step toward energy awareness. At a minimum, it provides a baseline to compare when introducing changes into the system. Costly additions, in terms of energy, may need to be revised or even discarded if low power consumption is a priority.

Different methods for measuring energy consumption exist. Because software is often dynamic and depends on runtime information, such as the size of exchanged data, understanding the trend of energy efficiency is more important than knowing the raw logged values. Just like the Big-O notation in time

ABOUT THE AUTHORS

ALCIDES FONSECA is an assistant professor at LASIGE, Faculdade de Ciências da Universidade de Lisboa, Portugal. Contact him at alcidesfonseca@gmail.com.

RICK KAZMAN is a professor at the University of Hawaii, Mãnoa, Honolulu, and a visiting scientist at the Software Engineering Institute, Pittsburgh, Pennsylvania. Contact him at kazman@hawaii.edu.

PATRICIA LAGO is a professor in software engineering at the Vrije Universiteit Amsterdam, The Netherlands. Contact her at p.lago@vu.nl.

complexity, energy efficiency can be seen as another complexity metric.

We believe that energy-aware ICT is inevitable for many reasons: economic reasons, sustainability reasons, and because users will increasingly demand it. To hasten its emergence, we have polled the R&D community in a series of workshops and collected these nine principles to help focus the emergence of energy awareness as a true subdiscipline of ICT and software engineering. We can see that the successful emergence of this discipline depends on three foundations: 1) awareness, 2) education and training, and 3) the creation of a body of engineering knowledge. We need all of them to make energy-aware software a reality.

References

1. "'Tsunami of data' could consume one fifth of global electricity by 2025," *The Guardian*, Dec. 11, 2017. [Online]. Available: https://www.theguardian.com/environment/2017/dec/11/tsunami-of-data-could-consume-fifth-global-electricity-by-2025
2. "6th International Workshop on Green and Sustainable Software." [Online]. Available: http://greens.cs.vu.nl
3. C. Becker et al., "Sustainability design and software: The Karlskrona manifesto," in *Proc. 37th Int. Conf. Software Engineering*, vol. 2, pp. 467–476.
4. G. Pinto and F. Castor, "Energy efficiency: A new concern for application software developers," *Comm. ACM*, vol. 60, no. 12, pp. 68–75, Dec. 2017.
5. N. Condori-Fernandez and P. Lago, "Characterizing the contribution of quality requirements to software sustainability," *J. Syst. Softw.*, vol. 137, pp. 289–305, Mar. 2018. doi: 10.1016/j.jss.2017.12.005.
6. "Peter Drucker," Wikipedia. Accessed on: Sept. 16, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Peter_Drucker
7. C. Pang, A. Hindle, B. Adams, and A. E. Hassan, "What do programmers know about software energy consumption?" *IEEE Softw.*, vol. 33, no. 3, pp. 83–89, 2016. doi: 10.1109/MS.2015.83.

Digital Object Identifier 10.1109/MS.2019.2944045