# Demeter: An Architecture for Long-Term Monitoring of Software Power Consumption

Lylian Siffre, Gabriel Breuil, Adel Noureddine, Renaud Pawlak

# Demeter: An Architecture for Long-Term Monitoring of Software Power Consumption

Lylian Siffre[1], Gabriel Breuil[1], Adel Noureddine[2[0000−0002−8585−574X]], and Renaud Pawlak[3]

[1] Constellation, France
lylian.siffre@impakt.io, gabriel.breuil@constellation.fr
[2] Universite de Pau et des Pays de l'Adour, E2S UPPA, LIUPPA, Pau, France
adel.noureddine@univ-pau.fr
[3] Cinchéo, Paris, France
renaud.pawlak@gmail.com

**Abstract.** Quantifying and long-term monitoring the energy consumption of software in end-user computers is a complex task. It brings multiple technical and sociological challenges. End users need to visualize their energy consumption and get a per-software feedback about their energy impact to adapt their software usage towards a greener approach. In this paper, we present our monitoring and feedback architecture: Demeter. Our distributed approach monitors energy consumption per application on runtime, provides end users with immediate feedback through a graphical user interface, and delayed feedback through an analysis email notification. We illustrate our approach with a two-week study of software usage of three different user profiles in a corporate environment.

**Keywords:** Power Monitoring · Measurement · Energy Consumption · Long-term monitoring · Distributed Architecture · Software Engineering

## 1 Introduction

Information and Communications Technology (ICT) has both direct and indirect impacts. The direct impacts are the production, use, and disposal of hardware while the indirect impacts are the effects of use (induction and obsolescence) and the extended effects (rebound and emerging risks) [5]. The indirect impacts are due to the abstract representation of software, API, browsers, networks, virtualization, and the offshoring of the manifold data centers over the world [1, 6]. Thus, one has to evaluate how much is ICT energy consuming and what are the solutions to reduce its energy consumption. The impact of ICT is already at 4% of the total worldwide energy consumption, more than civil aviation [4]. It is expected that in 2030 the energy consumption of ICT will triple [10].

Energy consumption monitoring software already exists with basic ones such as the task manager, available on modern OS, informs qualitatively on the software energy consumption. Unfortunately, the given data are not always accurate nor specific. In the past ten years, scientists and practitioners have focused on

developing more elaborate energy-probing software. Johann *et al.* [9] discussed the importance of using a generic metric to quantify energy consumption. They suggest two methods: 1) a white-box method consists in measuring the energy consumption of the source code and 2) a black-box method consists in measuring the energy consumption of the whole software. Thus the latter allows one to perform benchmark and individual measurements on computers. Two major issues arise from the existing energy probing software: 1) they only quantify the energy consumption of a specific group of software [3, 8, 12] or of the general energy consumption of a computer without software specification, and 2) the probing software is not hardware specific. Even though the CPU is well known to be energy consuming, it is essential to consider the energy consumption of every hardware component [2, 8, 12].

Most recent monitoring software at the component and application levels, such as Jolinar and PowerJoular [7, 11], are capable of monitoring hardware components for a limited number of applications (typically one application or process), and are primarily available on server environments or Linux systems. In particular, existing monitoring solutions are either limited in capabilities (*i.e.*, only monitoring specific hardware components or a particular software), or cannot scale up for multi-days or weeks of monitoring (*i.e.*, huge data collected, high CPU or energy impact of the monitoring tool, invasive monitoring interface, etc.). Moreover, end users tend to be less savvy and more reluctant to install monitoring software on their computers. The difficulty is in convincing users and administrators alike to deploy a monitoring software. Our goal is summarized in two main objectives: 1) provide a long-term monitoring software (days and weeks measurements), and 2) provide energy feedback to users with fine granularity (per application and per hardware component). Our presented software architecture, Demeter, implements these two objectives and allows practitioners, researchers, and industrial managers to study the energy impact of devices and users, and encourage eco-friendly software usage behavior.

In this article, we first present Demeter's architecture and its energy consumption models in Section 2. Then we present, in Section 3, a use case scenario using Demeter and aiming towards studying the energy impact of users' software usage in a corporate environment. Finally, we conclude in Section 4.

## 2 Architecture of Demeter

In this section, we present Demeter's architecture in figure 1. The intended purpose is to develop a probing software as global as possible regarding the energy consumption of applications.Demeter interacts with the end user (who can view its power consumption and get immediate or delayed feedback), and with applications and the OS to collect data for usage and energy monitoring. We decided to focus our attention on four criteria: *simplicity* in installation, programming, and usage, *energy efficiency* as Demeter must have the lowest energy consumption in regards to other applications, *adaptability* is the possibility to choose the

probing frequency and change the conversion values, and *lightness* to have a low number of external libraries and light-weighted output files.

Figure 1 presents the container diagram of Demeter with its three main parts:

- Monitoring Application (MA): responsible for the usage and power monitoring of every application and hardware component. It collects data from running applications and the OS. Provides the results to the graphical user interface and to the reporting server.
- Graphical User Interface (GUI): a graphical interface software allowing per-application visualization to end users, aggregated statistics, and historical power usages.
- Reporting Server (RS): responsible for analyzing long-term power data, providing weekly summaries and notifications to end users about their energy usage and impacts.

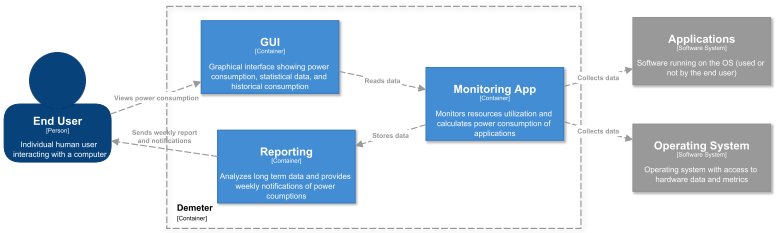


Fig. 1: Container diagram of Demeter architecture

Figure 2 presents the component diagram of the Monitoring Application. Each hardware component has a dedicated software component implementing the relevant sensors to collect usage or power data, and its associated power formulas and models to estimate its power consumption. An additional component is also implemented to collect processes and applications information and usage, and a utility component manages the input/output of the application and communications with the Remote Server. Demeter's architecture is OS-agnostic, but our initial implementation only targets Microsoft Windows as it is the most used desktop OS in corporate environments.

To satisfy the *simplicity* criterion, we decided to develop a one-agent probing software. Thus a non-Object-Oriented Programming (OOP) approach seems to be more suitable, reducing the memory footprint of Demeter and the cost of object creation and management. Motivated by making Demeter's energy footprint as low as possible, for the *energy efficiency* criterion, we programmed in C++ since it is ranked among the most efficient programming languages [13]. Each data source is handled in an independent file, thus no side effect on the software can be expected when adding or removing a data source. The behavior of Demeter is based on a unique periodic `while` loop for which the time between

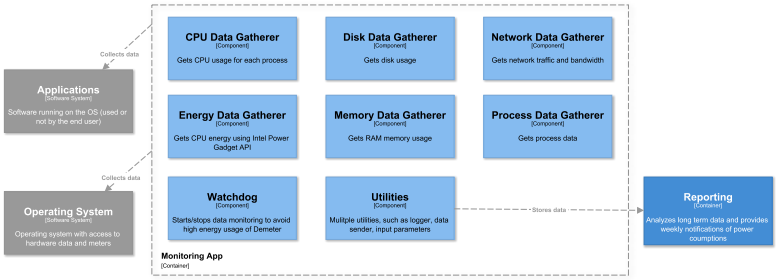


Fig. 2: Component diagram of Demeter's Monitoring Application

two iterations of this loop is at least an interval long. It will sequentially collect data from all activated sources (*i.e.*, calls to a function). Gathering memory and disk data is done through calls to the Windows Win32 API [4].

**CPU:** In Demeter, the energy consumption is collected through Running Average Power Limit (RAPL) and Intel Power Gadget (IPG). They are model-based software and are precise enough to rely on [14]. We define the CPU percentage usage of a process as in Equation 1.

$$U = (\frac{t_{\text{user}} + t_{\text{kernel}}}{t_{\text{total}}})/n_{\text{cores}} \tag{1}$$

Where $n_{\text{cores}}$ is the CPU's core amount, $t_{\text{user}}$ is the duration during which the process runs in user mode, $t_{\text{kernel}}$ is the duration during which the process runs in kernel mode and $t_{\text{total}}$ is the duration during which the CPU is active and idle. These values are retrieved from PSAPI [5]. In Equation 1, the sum between $t_{\text{kernel}}$ and $t_{\text{user}}$ corresponds to the invested CPU time for a process. To avoid having values above 100%, it is required to divide by $n_{cores}$.

**Hard Drive:** We use the read and write performance power ratio to calculate the disk energy consumption, in addition to the idle power consumption. This ratio is usually provided by manufacturers of modern disk drives. In addition, if the sequential read and write performance power ratio is also given, we take this ratio into consideration as it is more aligned with disk operations.

**Network:** Our aim is to probe the energy consumed by the network interface controller as Demeter will only estimate the energy consumed by the computer and its components. Gathering the bandwidth for every process is done using the Npcap library [6]. Npcap sniffs and reads the content of every packet passing through a network interface. Every available network interface of the computer is then opened and sniffed. For each process, the upstream and downstream bandwidths are gathered by our packet parser, using Npcap. The network energy

[4] https://learn.microsoft.com/en-us/windows/win32/api/
[5] https://learn.microsoft.com/en-US/windows/win32/psapi/psapi-functions
[6] https://npcap.com

consumption of a given software is the sum of its processes' energy consumption. We base our model on data from manufacturers' Ethernet transceiver's data sheet.In particular, most transceiver can negotiate multiple bitrates, such as 10/100/1000 Mb $\cdot$ s$^{-1}$, each with its own energy consumption.

**WatchDog:** Demeter has to control its CPU usage to have the smallest impact on the system. Our CPU consumption regulator is called WatchDog (WD), which can be activated or deactivated by the user. It will pause Demeter's activity if it becomes too CPU-consuming. The calibration of the WD is based on the CPU average consumption of Demeter over the first hour. It pauses the software for one minute if it detects an over-consumption (*i.e.* if its energy consumption is higher than three times the energy mean value during the calibration hour).

**Immediate Feedback through a Graphical User Interface (GUI):** The GUI (shown in Figure 3) aims to give immediate feedback to users as Demeter is also meant to be used by non-technical end users. The GUI allows the user to create their own dashboard by adding, moving, and removing graphics. The graphics can be configured with two main parameters: the program to plot and the resource to plot. The user has the possibility to display its real-time energy consumption, the cumulative energy consumption of an hour of the day, and the cumulative energy per day of the current week.

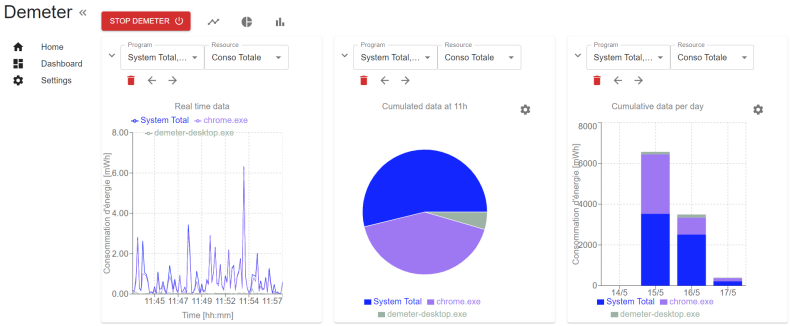


Fig. 3: Screenshot of Demeter's GUI

The GUI allows end users to monitor the power consumption of all applications. For each application, users can monitor real-time power consumption, the historical power evolution (through time), and aggregated statistics about power usage. The flexibility of the GUI (per-application and system-wide measurements and statistics), allows end users to customize the interface according to their desires and needs, thus allowing a personalized approach with the tool and in regards to power consumption. We argue that such a relationship where the user feels empowered in freely and easily choosing what to follow might lead to better engagement with power efficiency software usage best practices, and thus to an overall reduction of power consumption.

**Delayed Feedback through Cloud Notifications:** The third part of Demeter is the Remote Server (RS). The server component is responsible for: 1) storing all power data sent by the Monitoring Application for all related users (for instance, one RS can handle all users in an office, a company, or a building), 2) analyzing per-user and cross-user power consumption patterns and trends and providing an overview of a population's power profiles (for example, through a dashboard), and 3) sending recurrent notifications to users with a summary of their power consumption along with trends and power evolution, and with recommendations for power reductions. For instance, an email notification could be sent to users in a corporate environment with the power analysis of their consumption of the prior week.

In the next section, we present a real-world experiment scenario using Demeter to monitor users software and energy usage in a corporate environment.

## 3 Long-Term Monitoring Preliminary Study of Software Energy Consumption in a Corporate Environment

We present in this section the effect of the usage on the energy consumption of applications. We first determine good practices on the energy consumption of applications, then we examine their impacts during a two-week experiment on the overall energy consumption and on user behavior. For our study, we implement the Monitoring Application and the GUI, but we did not implement the Remote Server.

### 3.1 Green Good Practices

To determine precise energy consumption good practices, we measure the energy consumption of specific application usages on a laptop[7]. We evaluate the energy savings that can be made when reducing the quality of a YouTube video, reducing the quality of music streaming on Spotify, having the lowest possible number of opened tabs on Google Chrome, and deactivating the camera on Teams.

*Watching YouTube videos:* All running applications were closed except Demeter and a YouTube web page on Chrome. We launched the first 30 minutes of the following YouTube video [8]. We measured the energy consumption of Chrome for three video resolutions: 160p (E = 27.445 mWh), 480p (E = 28.425 mWh), and 4K (E = 211.773 mWh). We observe that the energy consumption related to the video resolution at 160p and 480p are in the same order of magnitude while the energy consumption at 4K is 7.7 and 7.45 times higher, respectively. Since we do not measure the energy consumption of the GPU, we underestimate the overall energy consumption.We conclude that reducing the video resolution from 4K to 480p helps in reducing energy consumption.

[7] Dell Latitude 5420

[8] youtube.com/watch?v=XVkADAwOXnU, accessed 04/26/2023

*Music streaming:* All applications have been closed except for Demeter and Spotify. We played each of these two tracks once for 30 minutes [9, 10]. We measured the energy consumption of Spotify for two different audio quality: low ($24\,\text{Kbits.s}^{-1}$, $\text{E} = 7.563\,\text{mWh}$) and very high ($320\,\text{Kbits.s}^{-1}$, $\text{E} = 12.141\,\text{mWh}$). We conclude that decreasing the quality helps in reducing the overall energy consumption of Spotify of $4,578\,\text{mWh}$.

*Browsing webpages on Chrome:* We measured and compared the energy consumption of two different scenarios: 1) every minute we refresh one tab, in a round-robin cycle, while the 14 others are idle; and 2) every five minutes we open the tab of Wikipedia's random page [11] and close the previously browsed ones. We did both measurements during 30 minutes each. We observe that scenario 1 consumes $\text{E} = 30.649\,\text{mWh}$, and scenario 2 consumes $\text{E} = 17.993\,\text{mWh}$. Therefore, we saved $12,656\,\text{mWh}$ thanks to the scenario 2.

*Videoconferencing on Teams:* We started two 30-minute meetings on Teams, one with the cameras of both users on, and the one with cameras off. The experiment during which the cameras were activated had an overall energy consumption of $\text{E} = 1\,123.405\,\text{mWh}$ while the latter consumed $\text{E} = 164.059\,\text{mWh}$. Therefore, the deactivation of both cameras allowed a decrease in energy consumption by a factor 6.8 and we saved $\Delta\text{E} = 959.346\,\text{mWh}$.

### 3.2 Impact of the Good Practices During a Two-Week Experiment

We showcase the importance of our approach in a two-week experiment, studying the impact of the recommended good practices on the energy consumption and sustainability awareness of 3 corporate users. We run Demeter for two weeks on three laptops (Dell Latitude 5420) for 3 different user profiles: user 1 is a project leader in digital marketing, user 2 is a help desk technician, and user 3 is a researcher in Green IT. The three users performed their day-to-day tasks at work as usual. During the first week, no specific instructions have been given to the users. Then, users were briefed about sustainability software good practices before the second week. Users finally had to report how they used Teams, Spotify, and Chrome during the two weeks.

In Figure 4, we gather the energy consumption per day and per component for users 1, 2, and 3, and for Chrome and Teams. We observe that the CPU consumption embodies the total energy consumption. The CPU energy is higher than 99.998% of the total energy for both applications. The upload and download stream is lower than 0.068% of the total energy. The reading and writing phase of the hard disk is the least consuming part of the laptop since their energy consumption is lower than 0.002%.

[9] open.spotify.com/track/2QJx8IgKSFfbMQDKxMUioZ?si=13b3e7896d82491e
[10] open.spotify.com/track/3KtsRijwp8KunCRYlOdWEi?si=373f58f7accb47ba
[11] https://en.wikipedia.org/wiki/Special:Random

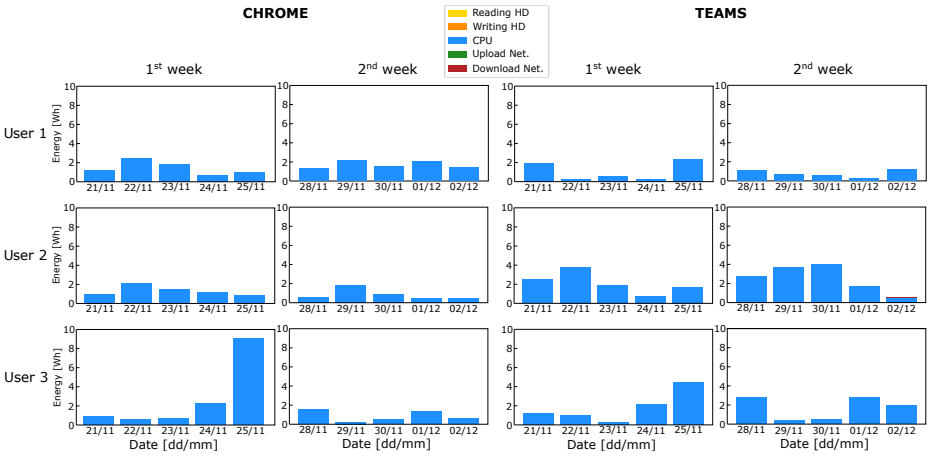


Fig. 4: Energy consumption [Wh] per component (the reading phase of the hard disk is yellow, the writing phase of the hard disk is orange, the CPU is blue, the upload stream is green, and the download stream is red) of Chrome (on the left) and Teams (on the right) during two weeks for users 1, 2, and 3.

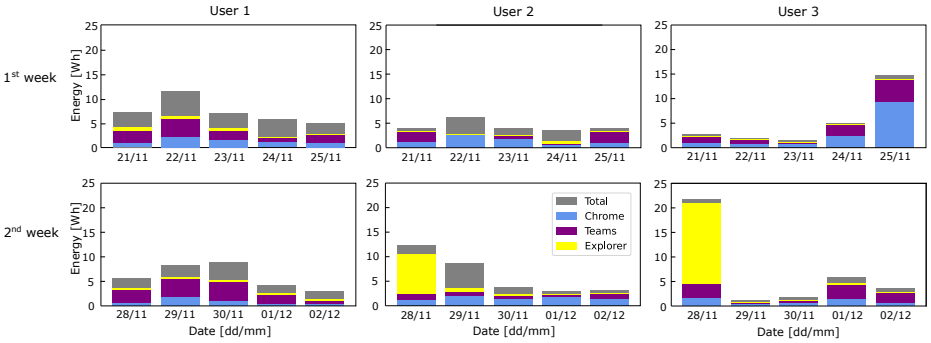


Fig. 5: Energy consumption [Wh] per application (Explorer is yellow, Teams is purple, Chrome is blue, and the total energy consumption of the computer is gray) during two weeks for users 1, 2, and 3.

*Total:* Figure 5 shows the energy consumption of users 1, 2, and 3 during two weeks, with the total energy of all applications (in gray), of Chrome (in blue), of Teams (in purple), and of the Windows Explorer (in yellow, a file manager in Windows). Teams and Chrome represent a significant part of the overall energy consumption of a computer for these three users. In the first week, the sum of both applications is 57% (User 1), 48% (User 2), and 90% (User 3) of the overall energy consumption while in the second week, they represent 40% (User 1), 57% (User 2), and 37% (User 3) of the overall energy. Since Teams was less solicited for User 1 (no camera) during the second week, a reduction of energy is detected.

During the first week, Teams represent 24% of the total energy consumption while it represents 12% in the second week. Users 2 and 3 used Teams more during the second week which lead to an increase in energy consumption. It represents 30% (User 2) and 35% (User 3) of the overall energy in the first week while it represents 43% (User 2) and 25% (User 3) in the second. Finally, we see for users 1 and 3 that on 28/11 there is an energy consumption peak, $E_{\text{tot}} = 12.35\,\text{Wh}$ (User 1) and $E_{\text{tot}} = 21.76\,\text{Wh}$. (User 3). This is due to the usage of Windows Explorer, while its energy consumption is very low during the other days. As we didn't collect specific and detailed software usage (through our tool or the questionnaire), it is difficult to analyze why we have this peak for a specific application on 28/11. However, Demeter allows users and system administrators to get an insight into software energy consumption for multiple days and weeks, and therefore identify energy leaks or abnormal energy behavior.

### 3.3 Discussions and limitations

We recommended good practices for users to modify their software behavior in order to evaluate their impact on the application's energy consumption. All of the recommendations were simple to follow for all three users and did not require significant changes in their workflows. However, some actions were more bothering for users, such as closing the browser's tabs. Since users are not used to frequently close their tabs, they had to constantly be aware and remember they have to close unused tabs. This constant awareness might either fade and users stop applying the recommendations, or it might turn into a habit. A similar conclusion is brought on the deactivation of the camera which is not a systematic habit for users. Moreover, this action involves other users who would like to keep their camera activated or wish to see the face of the speaker. We also found that the two easiest recommendations to follow were reducing the music and the video quality. Both recommendations required some actions at the beginning (such as changing YouTube's parameters). A longer experiment might give us a wider overview of the impact of good practices recommendations on the software's energy consumption and on user behavior.

Our study and our approach have a few limitations: 1) our study only consisted of three users. Although different in their user profile and role in the office, they all worked for the same company, 2) the experiment lasted for two weeks, with the first week serving as a control study, and the second aimed to study the impact of the green recommendations. Although we monitored software energy constantly for two weeks, we argue that a complete study needs to analyze usage for longer times, and 3) our study is the first essay into understanding user software usage in an office environment, and therefore we did not follow field studies protocol (no control group, no randomization of groups, low number of participants, etc.). However, we tried to control the experiment with a week of normal usage, and we aimed for three different user profiles.

## 4 Conclusion and Perspectives

We presented Demeter, a software architecture capable of energy consumption long-term monitoring of software. We conducted a real world experiment with three users in a corporate environment for two weeks. We observed a significant difference in energy consumption after providing users with green software good practices. We aim to conduct a multi-month study of multiple users in a large corporate environment using Demeter, with a goal to provide insights into their energy patterns, impact, and acceptance of green recommendations and actions.

## References

1. Bieser, J.C.T., Hilty, L.M.: Assessing indirect environmental effects of information and communication technology (ict): A systematic literature review. Sustainability **10**(8) (2018)
2. Chen, H., Wang, S., Shi, W.: Where does the power go in a computer system: Experimental analysis and implications. In: 2011 International green computing conference and workshops. pp. 1–6. IEEE (2011)
3. Dick, M., Kern, E., Drangmeister, J., Naumann, S., Johann, T.: Measurement and rating of software-induced energy consumption of desktop pcs and servers. In: EnviroInfo. pp. 290–299 (2011)
4. Efoui-Hess, M.: Climate crisis: The unsustainable use of online video. The Shift Project pp. 1–36 (2019)
5. Hankel, A., Heimeriks, G., Lago, P.: A systematic literature review of the factors of influence on the environmental impact of ict. Technologies **6**(3), 85 (2018)
6. Horner, N.C., Shehabi, A., Azevedo, I.L.: Known unknowns: indirect energy effects of information and communication technology. Environmental Research Letters **11**(10), 103001 (oct 2016)
7. Islam, S., Noureddine, A., Bashroush, R.: Measuring energy footprint of software features. In: 2016 IEEE 24th International Conference on Program Comprehension (ICPC). pp. 1–4. IEEE (2016)
8. Jagroep, E., van der Werf, J.M.E., Jansen, S., Ferreira, M., Visser, J.: Profiling energy profilers. In: Proceedings of the 30th annual ACM symposium on applied computing. pp. 2198–2203 (2015)
9. Johann, T., Dick, M., Naumann, S., Kern, E.: How to measure energy-efficiency of software: Metrics and measurement results. In: 2012 First International Workshop on Green and Sustainable Software (GREENS). pp. 51–54. IEEE (2012)
10. Jones, N.: The Information Factories. Nature **561**, 163–166 (2019)
11. Noureddine, A.: Powerjoular and joularjx: Multi-platform software power monitoring tools. In: 18th International Conference on Intelligent Environments (2022)
12. Ournani, Z.: Software eco-design: investigating and reducing the energy consumption of software. Ph.D. thesis, University of Lille (2021)
13. Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J.P., Saraiva, J.: Ranking programming languages by energy efficiency. Science of Computer Programming **205**, 102609 (2021)
14. Rotem, E., Naveh, A., Ananthakrishnan, A., Weissmann, E., Rajwan, D.: Power-management architecture of the intel microarchitecture code-named sandy bridge. IEEE Micro **32**(2), 20–27 (2012)