Chapter 4
FEETINGS: Framework for Energy Efficiency Testing to Improve eNvironmental Goals of the Software

Javier Mancebo, Coral Calero, Félix García, Mª Ángeles Moraga, and Ignacio García-Rodríguez de Guzmán

Abstract Energy consumption and carbon emissions caused by the use of software have been increasing in recent years, and it is necessary to increase the energy awareness of both software developers and end users.

The objective of this chapter is to establish a framework that provides a solution to the lack of a single and agreed terminology, a process that helps researchers evaluate the energy efficiency of the software, and a technology environment that allows for accurate measurements of energy consumed. The result is FEETINGS (Framework for Energy Efficiency Testing to Improve eNvironmental Goals of the Software), which promotes the reliability of capture, analysis, and interpretation of software energy consumption data.

FEETINGS is composed of three main components: an ontology to provide precise definitions and harmonize the terminology related to software energy measurement; a process to guide researchers in carrying out the energy consumption measurements of the software, and a technological environment which allows the capture, analysis, and interpretation of software energy consumption data.

In addition, an example of the application of FEETINGS is presented, as well as a guide to good practice for energy efficiency of software, based on different experiments carried out with this framework.

The results obtained demonstrate that FEETINGS is a consistent, valid, and useful framework to analyze the energy efficiency of software, promoting the accuracy of its energy consumption measurements. Therefore, FEETINGS serves as a tool to make developers and users aware of the impact that software has on the environment.

J. Mancebo · C. Calero (✉) · F. García · M. Á. Moraga · I. G.-R. de Guzmán
Alarcos Research Group, Institute of Technologies and Information Systems, University of Castilla-La Mancha (UCLM), Ciudad Real, Spain
e-mail: Javier.Mancebo@uclm.es; Coral.Calero@uclm.es; Felix.Garcia@uclm.es; MariaAngeles.Moraga@uclm.es; Ignacio.GRodriguez@uclm.es

C. Calero et al. (eds.), *Software Sustainability*,
https://doi.org/10.1007/978-3-030-69970-3_4

4.1 Introduction

Tablets, computers, smartphones, smartwatches, and a multitude of technological devices have invaded our daily lives. All of these devices require energy to operate, which has led to a huge annual growth in energy consumption. According to recent studies, energy used for global information and communications technology (ICT) could exceed 20% of total energy, and emit up to 5.5% of the world's carbon emissions by 2025 [1, 2]. These data on the growth of energy consumption and global emissions have raised issues of great concern for both software professionals and users.

Although hardware is generally seen as the main culprit for ICT energy usage, software also has a tremendous impact on the energy consumed [3]. Unfortunately, to date, little attention has been given to this topic by the information and communications technology (ICT) community [4]. However, in recent years, trends such as "Green Software" have gained importance [5]. The purpose of green software is to promote improvements in the energy efficiency of software, minimizing the impact it may have on the environment [6, 7].

To improve the energy efficiency of software, it is first necessary to raise energy awareness among all stakeholders [4]. On the one hand, developers must be aware of the energy that the software consumes when used, so that they can develop more energy-efficient and environmentally friendly software. And software professionals in general must treat energy efficiency as a quality attribute of the software, in the same way that usability or security is treated [8–10].

On the other hand, awareness also needs to be raised among end users as to how much energy is required by the software they use on a daily basis, so that they are aware of the impact that software can have on the environment [4]. Ideally, end users could compare the software applications that meet their needs, and choose the option that consumes the least energy, and should also know how a given software application can be used in a more efficient manner from the point of view of energy consumption.

In order to raise awareness among stakeholders or to develop a sustainable and environmentally friendly software product, it is first necessary to know the energy consumption induced by the software when it is running, since if the energy consumption is not measured, it cannot be managed [11, 12]. As the European Union report indicates [13], "the existence of a methodology for measuring the energy or CO2 of the ICT infrastructure is extremely important for this sector, as it will allow the development of much more robust estimates of the impact of ICT."

However, there is currently a lack of both knowledge and tools to reliably and accurately analyze software energy consumption [5]. We consider that in this regard one can identify three main problems:

- Several inconsistencies and terminological conflicts appear [14] due to the fact that researchers have defined their methods of work using their own terms or concepts, provoking numerous examples of both synonymy (same concepts with different term associated) and homonymy (different concepts with the same

term). This lack of formal consensus makes it difficult to understand the main concepts involved when performing a software energy consumption assessment.
- There is a lack of a generally-agreed-on methodology that would guide software energy consumption assessments. This implies that the rigor of the studies carried out cannot be guaranteed, meaning that it is more complicated to replicate or compare the results obtained [15].
- Several measuring instruments are available for the analysis of software energy consumption. It is important to note that each measurement instrument has its own particular characteristics, and that it is necessary to choose the one that best adapts to the particular evaluation requirements concerned [16].

To contribute to the mitigation of these problems, and to be able to raise energy awareness among all stakeholders, we have developed a framework to promote the reliability of capture, analysis, and interpretation of software energy consumption data, known as "FEETINGS" (Framework for Energy Efficiency Testing to Improve eNvironmental Goals of the Software). FEETINGS aims to provide: (1) a solution to the lack of a unique and agreed terminology; (2) a process that helps researchers to evaluate the energy efficiency of the software, allowing greater control over the measurements made, thereby ensuring their reliability and consistency; and (3) a technological environment that supports the process and allows for realistic measurements of the energy consumed by the software and its subsequent analysis.

In this chapter, we present FEETINGS and an example of how to use it, so as to guide end users. The contents of the chapter are structured as follows. First, we present the different studies and proposals that have served as a basis for our framework. Then, in Sect. 4.2, we present the FEETINGS framework, detailing each of its components. In Sect. 4.3, an example of the application of FEETINGS is presented. In Sect. 4.4, a best practice guideline for software energy efficiency is proposed, based on different experiments carried out using FEETINGS. Finally, Sect. 4.5 sets out some conclusions of this work.

4.2 FEETINGS

In this section, we will describe FEETINGS, a framework to promote more reliable capture and analysis of software energy consumption data. This framework is made up of three main components, classified according to their nature as conceptual, methodological, and technological components (see Fig. 4.1), as described in the following subsections.

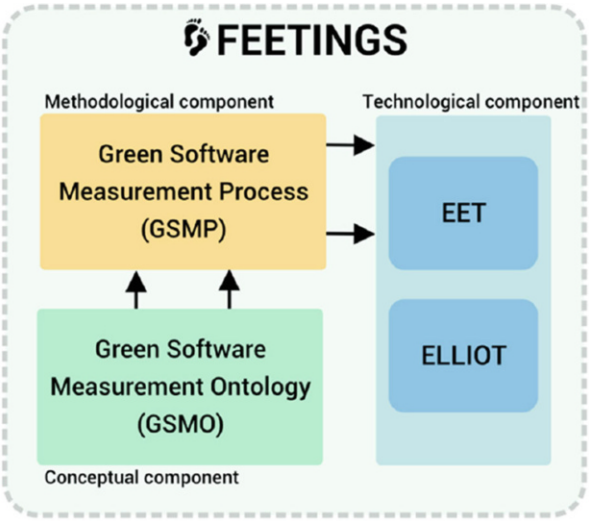

Fig. 4.1 Overview of FEETINGS

4.2.1 Conceptual Component

As commented upon in the introduction, one of the common problems is confusion and inconsistencies in the main concepts used in software energy assessment. This lack of formal consensus makes it difficult to understand the main concepts involved when performing a software energy consumption measurement.

The conceptual part of FEETINGS seeks to solve the lack of a unique and agreed terminology. For this purpose, an ontology has been elaborated which contains the concepts related to the software energy measurement. According to Chandrasekaran et al. [17], the unification of terms and concepts in an ontology allows knowledge to be shared, while ontological analysis clarifies the structure of knowledge.

The ontology proposed is known as "Green Software Measurement Ontology" (GSMO), and its purpose is to provide precise definitions of all terms related to software energy measurement and to clarify the relationships between them, removing terminological conflicts and fostering the consistent application of the framework by other researchers and practitioners with reference to a common vocabulary. The Green Software Measurement Ontology (GSMO) is an extension of the SMO ontology proposed by Garcia et al. [18] for green software measurement.

Figure 4.2 shows the graphical representation of the terms and relationships of the GSMO, using UML (Unified Modeling Language). The highlighted concepts are the new concepts which extend/adapt the SMO [18].

The conceptual component (GSMO ontology) aims to solve the problem of terminology consistency in software energy measurement, since it proposes a common vocabulary extracted from several international standards and research proposals. More details about the GSMO ontology can be found at [19].

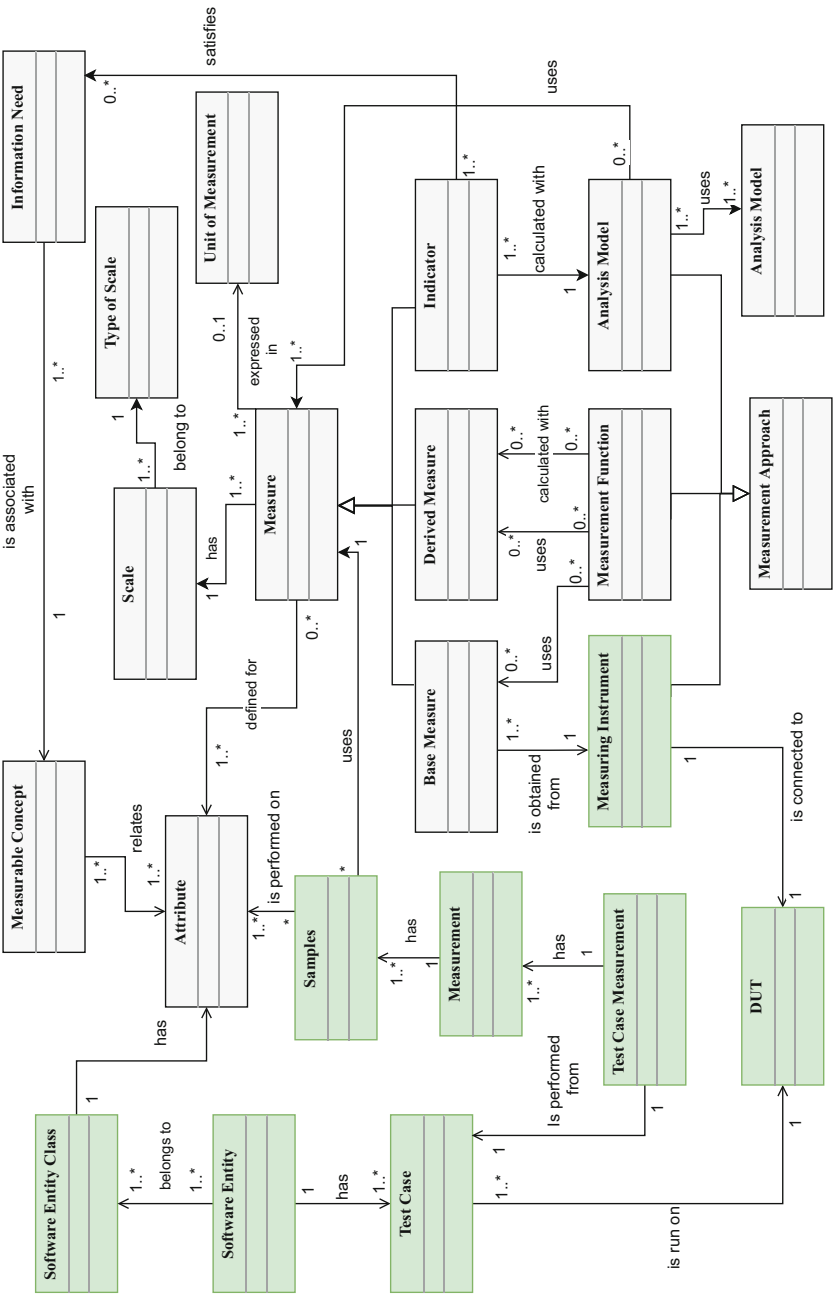

Fig. 4.2 UML diagram of the GSMO

This ontology has, moreover, served as a basis for the development of the methodological component of FEETINGS, which is presented in the following subsection.

4.2.2 Methodological Component

The methodological component consists in a process for measuring and analyzing the energy efficiency of the software. This process is known as the "Green Software Measurement Process" (GSMP). Its purpose is to guide researchers and practitioners as they seek to carry out measurements of software energy consumption. The GSMP ensures greater control over the measurements made, improving their reliability, consistency, and coherence. It also ensures that the results obtained are comparable with other studies and facilitates the replicability of the analyses performed.

To define the GSMP, we have followed the method engineering approach [20], and we have also taken as our basis well-known approaches to software measurement and good practices related to green software that have been proposed by other authors.

The process consists of seven phases, which are summarized below:

- *Phase I. Scope Definition:* In this phase, a complete specification of requirements for the evaluation of energy efficiency is obtained. In addition, the software which is to be the subject of the study and the test cases to be analyzed must be defined.
- *Phase II. Measurement Environment Settings:* The purpose of this phase is the definition of the measurement environment that is to be used in the software energy consumption assessment. As a result of this phase, the measuring instrument, its measurements, and the specifications of the Device Under Test (DUT) are defined and the baseline energy consumption of the DUT is obtained.
- *Phase III. Measurement Environment Preparation:* This phase focuses on the preparation of the energy consumption measurements to be performed and on the configuration of the measurement environment.
- *Phase IV. Measurements Performance:* During this phase, energy consumption measurements are carried out and raw energy consumption data taken from the measuring instrument is collected.
- *Phase V. Test Case Data Analysis*: The raw data of energy consumption obtained by the measuring instrument is processed, and the statistical analysis of the values obtained from the measurements of the defined test cases is carried out.
- *Phase VI. Software Entity Data Analysis:* In this phase, with the results obtained from the previous phases, the amount of energy consumed when the software entity was executed in the DUT is determined and interpreted, and some conclusions about the software energy consumption are stated.
- *Phase VII. Reporting the Results:* Finally, the study carried out is documented, describing the entire process followed, and setting out the results obtained on the energy consumption of the software.

In the previous chapter (Chap. 3) of this book, a more detailed and complete version of the GSMP is presented, including a description of the roles, phases, and activities (input, output, and guidelines).

4.2.3 Technological Component

In this section, the technological environment is presented. The main objective of the technological component of the FEETINGS framework is to perform more realistic measurements of the energy consumed by the software, and to use these results to

- Analyze the consumption of the software
- Learn about the behavior of the software and its different versions, to find out if these versions worsen the software energy consumption or not
- Identify the consumption patterns that can guide the improvement of the energy efficiency of software applications
- Recommend changes to software to improve energy efficiency

The technological component, as can be seen in Fig. 4.3, is composed of two artifacts: EET (Energy Efficiency Tester) and ELLIOT.

4.2.3.1 EET (Energy Efficiency Tester)

EET [21, 22] is a measuring instrument that enables the accurate capture of the energy consumption of the computer (DUT) on which the software is running. In addition to the total energy consumption of the DUT, this measuring instrument supports the measurement of four different hardware components: processor, hard disk, graphic card, and monitor. Figure 4.4 shows the EET measuring instrument in working use.

As can be seen in Fig. 4.4, the EET is connected to the DUT where the software is executed, and is composed of three main components:

- A system microcontroller, whose task is to gather the information extracted from the different sensors and store it in a MicroSD memory. It also allows the frequency with which the device performs the measurements to be adjusted.
- A set of sensors, which are responsible for taking energy consumption measurements of the hardware components (processor, hard disk, graphics card, and monitor) of the DUT connected to the EET.
- A power supply, which must be connected to the device under test where the software is executed, replacing the power supply of the DUT; the sensors are connected to the energy distribution lines from the power supply to the different hardware components.

In a nutshell, EET is a measuring instrument, which is considered a core component of FEETINGS. It allows us to capture and record the energy efficiency of

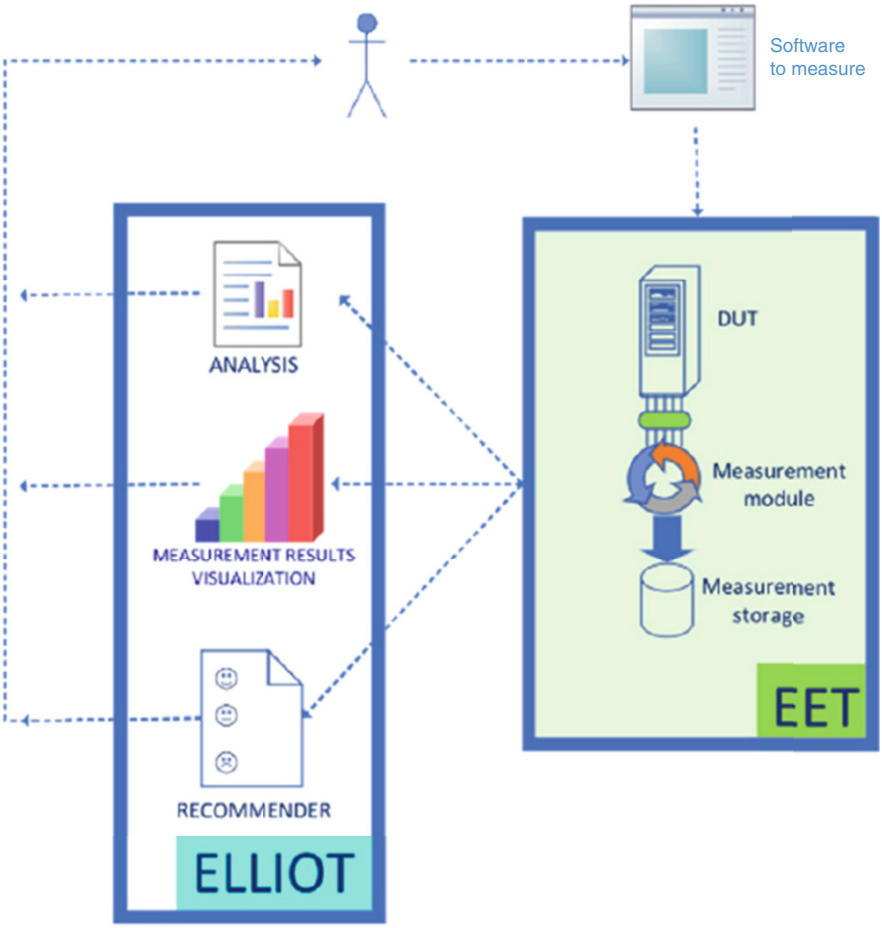

Fig. 4.3 Artifacts of the technological component of FEETINGS

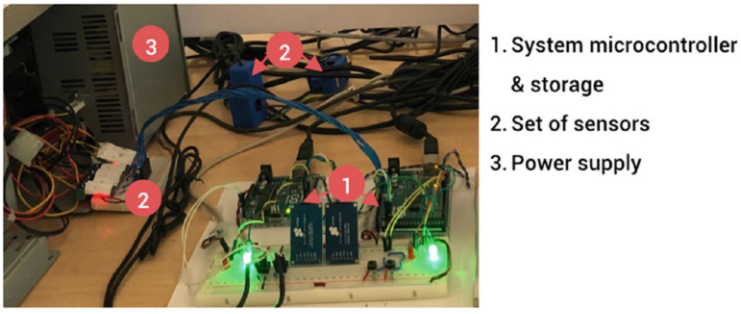

Fig. 4.4 EET measuring instrument

software when it is running. EET provides a realistic measurement of energy consumption and, moreover, is capable of obtaining detailed energy measurements from different components of the DUT (processor, graphic card, hard disk, and monitor). Another advantage of this measuring instrument is its sampling frequency, around 100 Hz, which provides very reliable consumption information.

As the EET produces a huge amount of data on energy consumption, it is necessary to support the processing and analysis of these data with a suitable software tool. For this reason, the ELLIOT tool was developed, which is described in the following section.

4.2.3.2 ELLIOT

ELLIOT [19] is a software tool tasked with processing the data collected by the EET, analyzing these data, and providing a visual environment that allows researchers to process the software energy consumption data. Furthermore, the ELLIOT tool is aligned with the GSMP described in the previous section.

The main functionalities supported by ELLIOT are outlined below:

- Processes all measurements carried out with the EET measuring instrument.
- Calculates different statistical variables of the energy consumption measurements according to the user's needs.
- Identifies possible outliers that may be present in the measurement samples, using robust parametric methods such as median absolute deviations from the median (MADN).
- Visualizes the results through graphs and data tables which contain information on the measurements of the energy consumption of the software.
- Compares the results obtained from the different energy consumption measurements.
- Generates reports that include all the information on the energy efficiency of the software analyzed.

The ELLIOT tool is composed of four modules (see Fig. 4.5) that support these functionalities. The modules are: (1) user management, which allows one to manage the permissions and roles of ELLIOT users; (2) system management, to add and modify information about the instruments and the DUT in which the measurements are carried out; (3) measurement management, which is the central module of ELLIOT since it supports all the tasks of processing, data wrangling, measurements analysis, and visualization of the energy consumption information; and (4) report management, which generates reports and allows comparisons between the measurements.

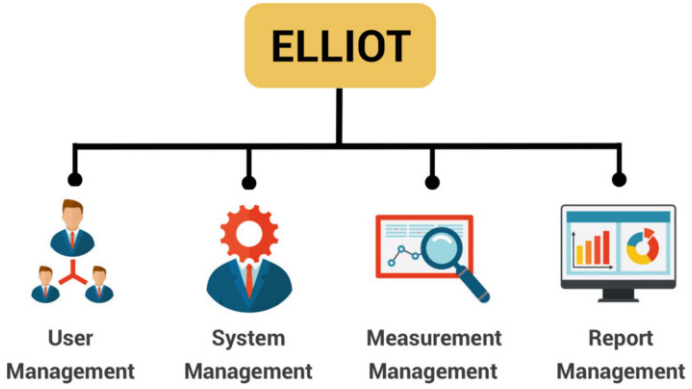

Fig. 4.5 ELLIOT tool modules

4.3 Application of FEETINGS: A Case Study of the Energy Consumed by Translators

This section presents an application of the FEETINGS framework to measure and analyze software energy consumption, as defined in the previous section. One of the most widely used software applications today is online translation, which includes, additionally, several options for automatic text translation, such as Google translate and DeepL.

In this study, our aim is to analyze the energy consumption of the main online translation tools in order to raise users' awareness of the environmental impact of their use, and also to try to provide them with a set of guidelines so that, when they use these tools, such use is as efficient as possible.

Following the GSMP, Phase I defines the scope of the study. As mentioned above, this study aims to determine the energy consumption of the main online translators (Software Entity Class). The chosen translators (Software Entity) were Google Translate, DeepL, Bing Translator, Tradukka, Systran Translate, and Yandex. To evaluate the selected software entities, five test cases were defined:

- Translate a text with 10 characters.
- Translate a text with 100 characters.
- Translate a text with 1000 characters.
- Translate a text with 3000 characters.
- Translate a text with 5000 characters.

All defined test cases were executed in two different browsers (Google Chrome and Firefox). Thus, we can also study the efficiency of the browser in which each of the translators is used.

In the second phase of the process, we selected the FEETINGS technological environment to analyze the energy consumption, choosing the EET as the measuring instrument and ELLIOT to analyze and process the energy consumption data. The

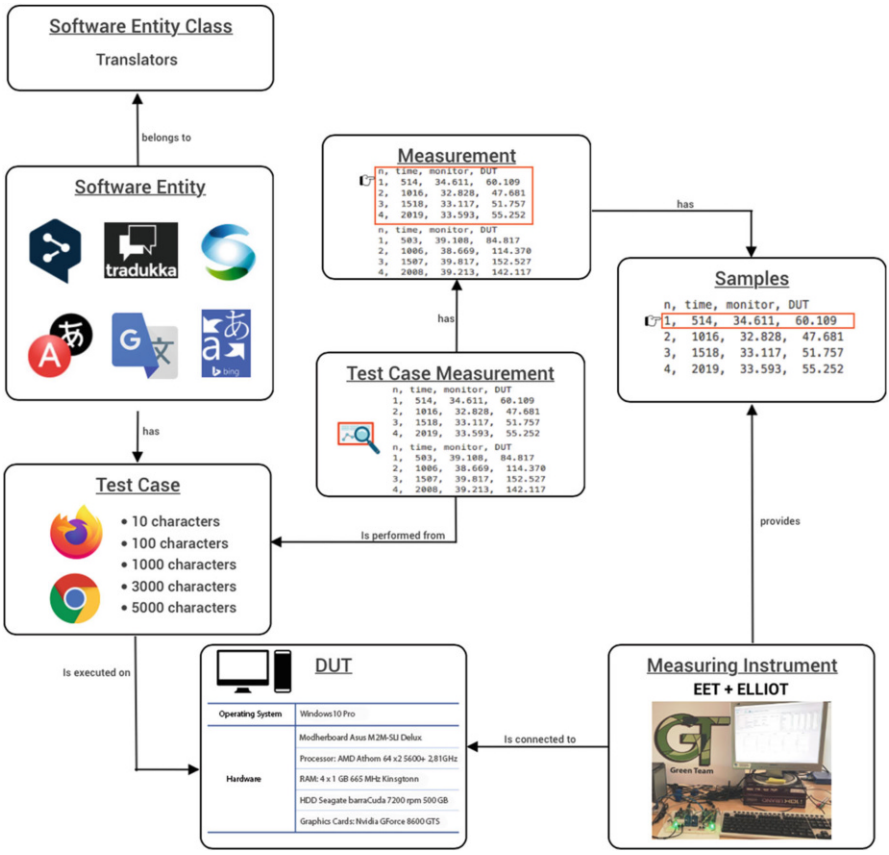

Fig. 4.6 GSMO instantiation

specification of the DUT in which the test cases were executed was also defined. For this study, we decided that from the measurements provided by the EET, we would take into account only the energy measurements of the monitor and the total consumption of the DUT. We have not recovered and analyzed data from the hard disk, graphics card, or processor because, as all the translators were executed in a web browser, the use of these components was minimal.

In accordance with the third phase of the process, we determined that each of the test cases was to be run and measured (with EET) 35 times. Being a controlled test environment, 35 measurements is usually a sufficient sample size to mitigate the impact of outliers (such as energy consumption devoted to operating system tasks).

Figure 4.6 shows the instantiation of the concepts of this study, defined in the GSMO ontology for this study, which also serves as a summary of the outputs obtained in the first phases of the GSMP.

Tables 4.1 and 4.2 show the energy consumed once the measurement and analysis tasks have been performed, by the DUT and the monitor respectively, in the execution of each of the defined test cases.

Table 4.1 DUT energy consumption for each test case

Test cases		DUT energy consumption (Watts per second)					
		DeepL	Bing	Google	Tradukka	Systran	Yandex
10 char.	Firefox	46.12	64.65	46.21	41.56	61.09	45.11
	Chrome	50.77	53.07	45.09	61.07	40.27	43.26
100 char.	Firefox	53.43	40.20	76.23	44.47	64.03	53.61
	Chrome	46.23	25.69	38.66	50.22	47.61	43.96
1000 char.	Firefox	57.34	39.83	52.25	54.12	59.72	66.52
	Chrome	48.91	28.99	45.42	32.57	40.67	51.37
3000 char.	Firefox	56.38	45.46	47.36	70.64	95.20	62.72
	Chrome	41.91	37.75	44.19	39.47	50.18	58.04
5000 char.	Firefox	67.31	57.62	49.45	62.98	77.29	66.56
	Chrome	51.77	52.53	45.30	41.53	56.05	69.93

Table 4.2 Monitor energy consumption for each test case

Test cases		Monitor energy consumption (Watts per second)					
		DeepL	Bing	Google	Tradukka	Systran	Yandex
10 char.	Firefox	63.06	70.40	65.54	53.27	65.50	60.14
	Chrome	66.88	56.41	63.92	68.34	57.62	59.96
100 char.	Firefox	75.83	58.51	62.28	61.07	64.77	70.15
	Chrome	61.23	50.64	59.34	57.48	57.09	61.17
1000 char.	Firefox	56.93	59.02	63.59	66.67	58.30	87.79
	Chrome	57.82	49.66	69.96	58.42	56.31	63.01
3000 char.	Firefox	56.61	54.90	67.10	52.87	73.70	72.07
	Chrome	53.35	52.97	66.95	56.43	60.38	64.14
5000 char.	Firefox	65.83	60.19	64.97	68.88	69.92	70.62
	Chrome	61.98	63.34	66.58	57.24	63.31	77.27

To analyze the data in the above tables, we will study the test cases executed in each of the browsers independently, and then compare the results obtained in both browsers.

First, focusing on the test cases executed in Firefox, in Fig. 4.7, we can observe that for translating texts of intermediate size (between 100 and 3000 characters) the option that requires the least energy consumption in the DUT is the Bing translator. However, Bing is the worst option for very small texts (around 10 characters), with Tradukka being the most efficient option in this case. For large texts (5000 characters) Google Translate is the best option. The consumption data results obtained from the monitor are very similar to those obtained from the DUT. The Yandex and Systran translators are the worst choice in almost every test case.

Analyzing the consumption data of the tests executed in Google Chrome, we can see in Fig. 4.8 that the energy consumption of the translators behaves in a similar way as in Firefox. In this case, Bing is also the best choice for intermediate-length text. But unlike the results obtained in Firefox, the Tradukka translator is the least efficient option for small texts (fewer than 100 characters).

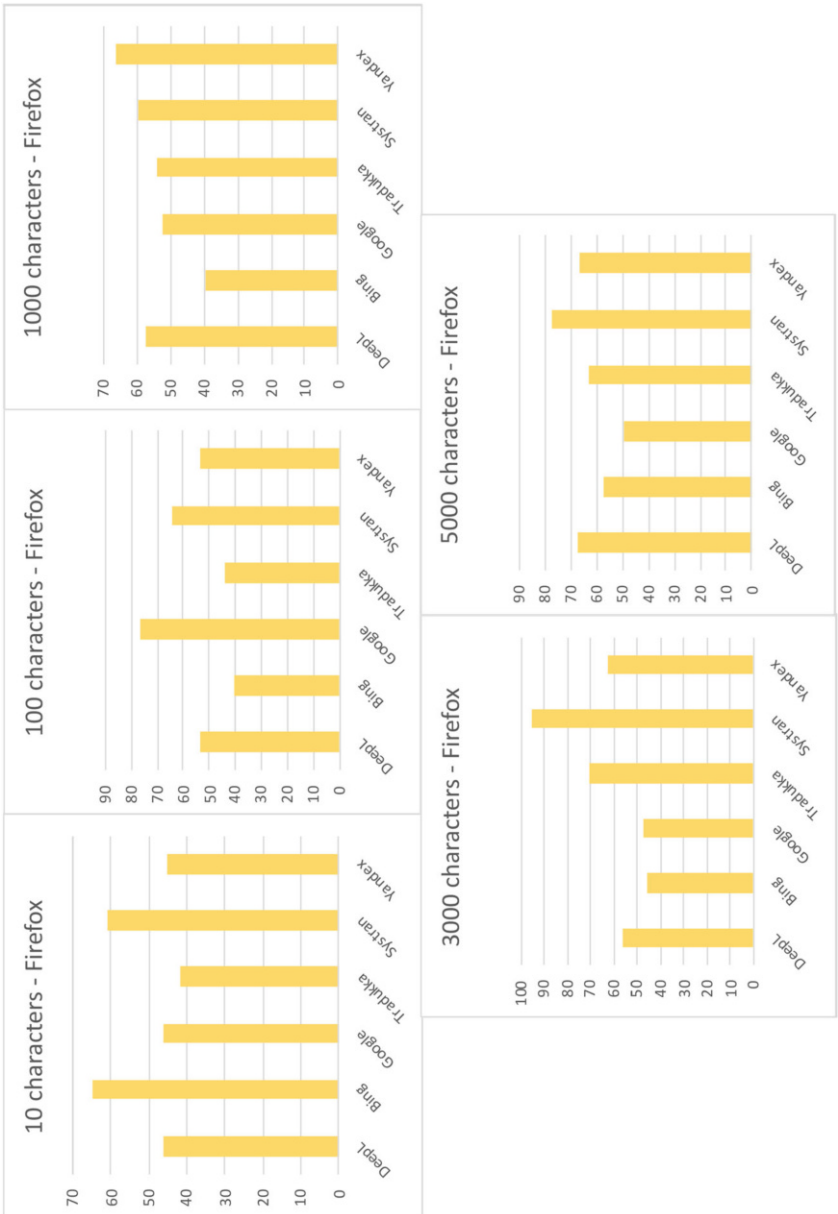

Fig. 4.7 DUT energy consumption for each test case in Firefox

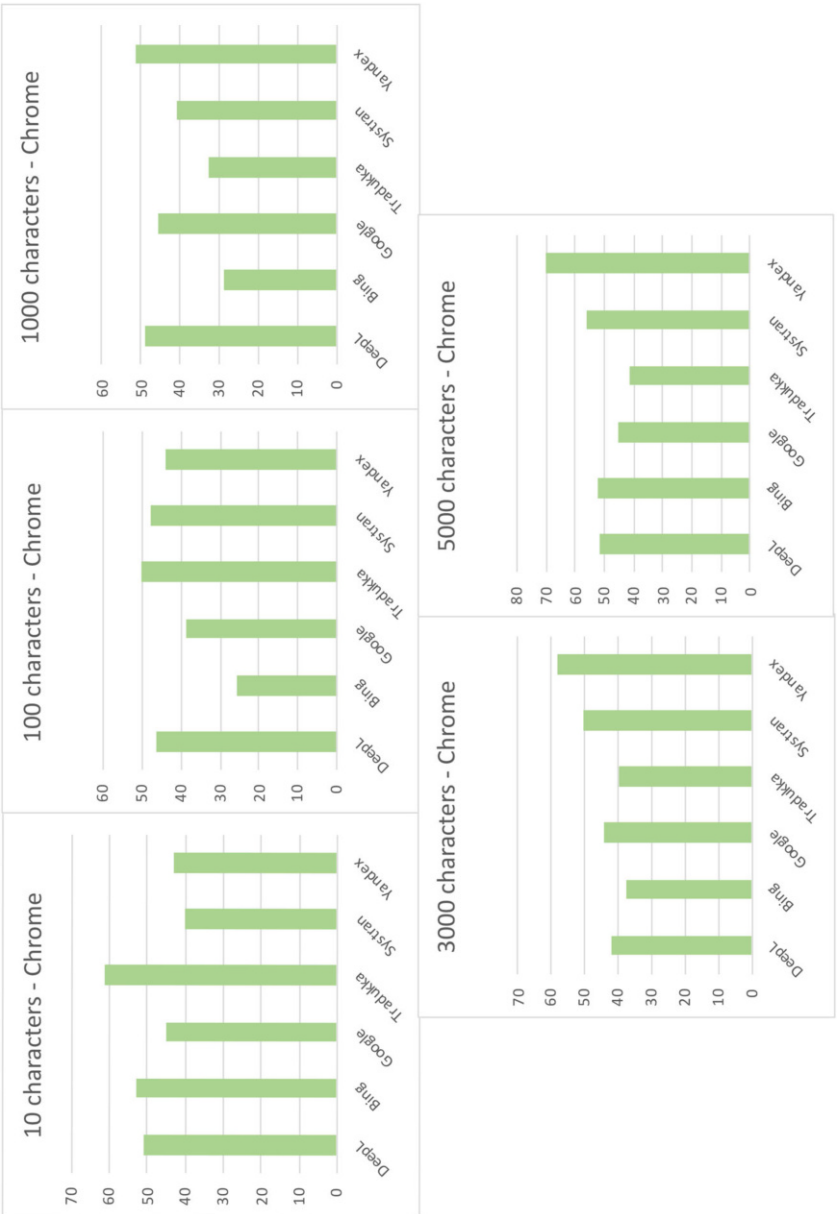

Fig. 4.8 DUT energy consumption for each test case in Chrome

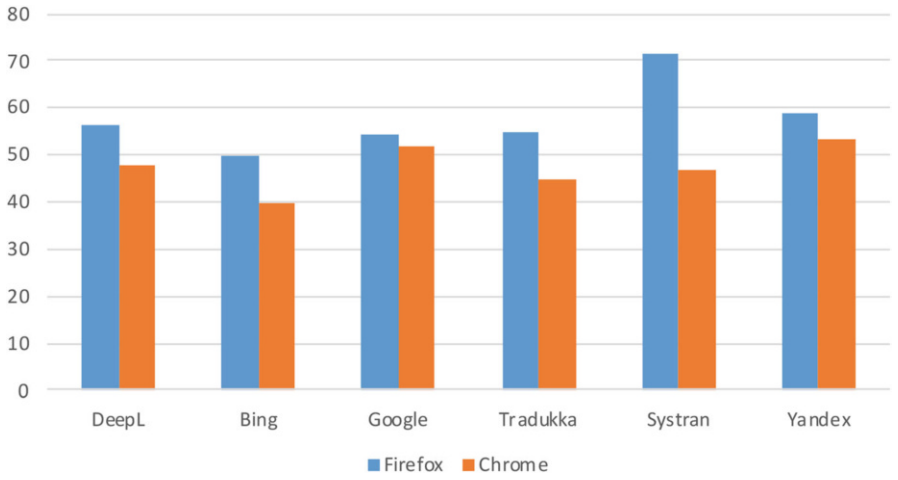

Fig. 4.9 Comparison of the mean energy consumption by the DUT

To determine in which browser (Firefox or Chrome) it is better to use the translators, we have calculated the average power consumption of the executed test cases. As shown in Fig. 4.9, all the translators analyzed have a lower DUT power consumption in the Chrome browser than in Firefox. Regarding the monitor's energy consumption, the results are similar to those of the DUT, although here the variation between browsers is less.

Considering the results obtained, the main conclusions that we can draw from this study are as follows:

- The most energy-efficient scenario is to use the Tradukka translator with the Google Chrome browser for texts of an intermediate length (1000 characters), resulting in a consumption of 32.56 Ws.
- The most inefficient option (95.19 Ws) is to use the Systran translator in the Firefox browser for large texts (3000 characters).
- All translators show a direct relationship when translating texts of more than 1000 characters, in which case an increase in the number of characters also increases energy consumption.
- It is more efficient to use the Google Chrome browser to translate than Firefox.

4.4 Best Practices Guideline on Software Sustainability

As explained in the introduction, FEETINGS can be used for several purposes. One of the main objectives of FEETINGS is to measure software energy efficiency so that researchers and software professionals can develop software that is environmentally friendly. Another purpose is focused on the other key perspective in software: the

end user. This framework can help to make society aware of the responsible use of software applications, so as to take care of the environment.

Keeping both perspectives in mind, this section presents some guidelines, based on several studies we have conducted, to make software more sustainable, in both its development and use.

First, we present the main findings that can be useful for researchers and practitioners wishing to develop software that is more energy-efficient:

- In the study presented in [23], it was concluded that the most efficient classification algorithm, in terms of energy, is the Quicksort, followed by the Bubble sort. In contrast, the most energy-demanding is the Insertion sorting algorithm.
- For the Redmine software [24], after analyzing different versions and the relationship between its energy consumption and maintenance measures, we can conclude that the Total Lines of Code (TLOC) maintainability measurement affects the energy consumption of the processor and the DUT.
- The study presented in [25] shows that text compression, using End-Tagged Dense Code (ETDC) or Tagged Huffman algorithms, not only reduces search space and time, but also leads to lower energy consumption. In addition, the use of search algorithms in compressed text, such as the Horspool algorithm, when run on compressed data, also requires less CPU power than when run over uncompressed data.
- In [26], recommendations were made to improve the energy efficiency of applications. The best practices in energy-efficient computing drawn from this study are summarized below:
  - A balance should be struck between the energy efficiency of the graphical user interface and a good experience from the user's perspective.
  - Efficient UIs should be designed to allow a task to be completed quickly and easily.
  - Data redundancy should be reduced.
  - Devices when not in use should be powered down by batch I/O.
- We have worked on the comparison of the time and energy consumption required by three releases of the same application, developed with and without Spring. In conclusion, it seems that products developed without using Spring are better in all the conditions and for all the measures. This could indicate that, although Spring has some advantages for programmers, once the product starts to run, this advantage disappears to the benefit of the non-Spring development.

Second, we present the following set of guidelines to help end users make use of the software in a more environmentally friendly way:

- To publish a tweet or a post on Facebook, you should consider that the most energy-efficient option is to publish a single emoji or a picture. Furthermore, if you want to be respectful of the environment, you should avoid publishing a GIF, since it is the option that consumes the most energy [19].

- The most energy-efficient personal health record (PHR) is the NoMoreClipboard [26].
- As regards navigating the internet: if you are looking for browsers which are environmentally friendly, choose Edge or Firefox; if you are looking for maximum privacy and energy efficiency use the DuckDuckGo search engine, especially when used with Edge and Firefox; if you are looking for lower emissions in your searches use the Ecosia search engine; and if in any event you do wish to use Chrome as your browser, then do so with DuckDuckGo.

4.5 Conclusions

Software plays an important role in the global energy consumption of a PC. For this reason, it is very important that both professionals and users be aware that the use of software has a great impact on the energy consumed by the devices on which it is executed.

In order to raise awareness of energy consumption among stakeholders, it is necessary to quantify its impact. Bearing this in mind, this chapter has presented the FEETINGS framework, which aims to promote reliable measurement, analysis and interpretation of software energy consumption data. FEETINGS is composed of three main components: (1) a GSMO ontology, to provide precise definitions of all concepts and their relationships related to software energy measurement; (2) a GSMP, to guide researchers in carrying out the energy consumption measurements of the software; and (3) a technological component, which is composed of two artifacts: EET, a measuring instrument, and the software tool ELLIOT to process and analyze data collected by the EET.

Thus, the use of FEETINGS serves two purposes. The first is to enable researchers and professionals to measure and make them aware of the energy that the software they develop consumes when in use, and thus be able to develop more energy-efficient software. The second is to show end users just how much energy is required by the software we use every day, and to make them aware of the impact that software can have on the environment.

In this work, we have also demonstrated an application of the FEETINGS framework to analyze the energy consumption involved in the use of different online translators. In addition, we have presented a set of best practice guides on sustainable software design based on our experience using the FEETINGS framework.

References

1. Andrae A (2019) Prediction studies of electricity use of global computing in 2030. Int J Sci Eng Invest 8:27–33

2. Vidal J (2017) Tsunami of data'could consume one fifth of global electricity by 2025. Climate Home News 11
3. Pereira R, Carção T, Couto M, Cunha J, Fernandes JP, Saraiva J (2020) Spelling out energy leaks: aiding developers locate energy inefficient code. J Syst Softw 161:110463
4. Fonseca A, Kazman R, Lago P (2019) A manifesto for energy-aware software. IEEE Softw 36 (6):79–82
5. Pinto G, Castor F (2017) Energy efficiency: a new concern for application software developers. Commun ACM 60(12):68–75
6. Calero C, Piattini M (2015) Introduction to green in software engineering. In: Green in software engineering. Springer, pp 3–27
7. Calero C, Piattini M (2017) Puzzling out software sustainability. Sustain Comput Informatics Syst 16:117–124
8. Calero C, Moraga MÁ, Bertoa MF, Duboc L (2014) Quality in use and software greenability. In: RE4SuSy@ RE. pp 28–36
9. Condori-Fernandez N, Lago P (2018) Characterizing the contribution of quality requirements to software sustainability. J Syst Softw 137:289–305
10. Penzenstadler B, Raturi A, Richardson D, Tomlinson B (2014) Safety, security, now sustainability: the nonfunctional requirement for the 21st century. IEEE Softw 31(3):40–47
11. Briand LC, Morasca S, Basili VR (1996) Property-based software engineering measurement. IEEE Trans Softw Eng 22(1):68–86
12. Moraga MÁ, Bertoa MF (2015) Green software measurement. In: Green in software engineering. Springer, pp 261–282
13. European Union (2011) The contribution of ICT to Energy Efficiency: local and regional initiatives
14. Pinto G, Castor F, Liu YD (2014) Understanding energy behaviors of thread management constructs. In: Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications, pp 345–360
15. Fenton N, Bieman J (2014) Software metrics: a rigorous and practical approach. CRC Press
16. Moura I, Pinto G, Ebert F, Castor F (2015) Mining energy-aware commits. In: IEEE/ACM 12th Working Conference on Mining Software Repositories. IEEE, pp 56–67
17. Chandrasekaran B, Josephson JR, Benjamins VR (1999) What are ontologies, and why do we need them? IEEE Intell Syst Their Applications 14(1):20–26
18. García F, Bertoa MF, Calero C, Vallecillo A, Ruiz F, Piattini M, Genero M (2006) Towards a consistent terminology for software measurement. Inf Softw Technol 48(8):631–644
19. Mancebo J, Calero C, García F, Moraga MÁ, García-Rodríguez De Guzmán I (2020) FEETINGS: Framework for Energy Efficiency Testing to Improve eNvironmental Goal of the Software. Paper presented at the The Eleventh International GREEN and Sustainable Computing (under review)
20. Henderson-Sellers B (2003) Method engineering for OO systems development. Commun ACM 46(10):73–78
21. Mancebo J, Arriaga HO, García F, Moraga MÁ, de Guzmán IG-R, Calero C (2018) EET: a device to support the measurement of software consumption. In: Proceedings of the 6th International Workshop on Green and Sustainable Software, pp 16–22
22. Piattini M, Calero C, García F, Moraga MÁ, de Guzmán IGR, Mancebo J, Arriaga HO, Tabaco R (2018) Aparato para medición del consumo eléctrico de equipos informáticos (PC). ES 1199234 Y

23. Mancebo J, Guldner A, Kern E, Kesseler P, Kreten S, Garcia F, Calero C, Naumann S (2020) Assessing the sustainability of software products—a method comparison. In: Advances and new trends in environmental informatics. Springer, pp 1–15
24. Mancebo J, Calero C, García F (2021) Does maintainability relate to the energy consumption of software? A case study. Softw Qual J 29(1):101–127
25. Mancebo J, Calero C, Garcia F, Brisaboa N, Fariña A, Pedreira O (2019) Saving energy in text search using compression. Paper presented at the GREEN 2019: The Fourth International Conference on Green Communications, Computing and Technologies, Nice, France
26. García-Berná JA, Fernández-Alemán JL, Carrillo-de-Gea JM, Toval A, Mancebo J, Calero C, García F (2020) Energy efficiency in software: a case study on sustainability in Personal Health Records. J Clean Prod