Review

Not peer-reviewed version

How to Improve Software Energy Efficiency? A Systematic Literature Review and the Current State of Applied Methods in Practice

Włodzimierz Wysocki * and Ireneusz Miciuła *

Posted Date: 30 January 2025

doi: 10.20944/preprints202501.2232.v1

Keywords: software; knowledge management; reasoning; information extraction; rules mining; knowledge acquisition and engineering

Article

How to Improve Software Energy Efficiency? A Systematic Literature Review and the Current State of Applied Methods in Practice

Włodzimierz Wysocki [1,*] and Ireneusz Miciuła [2]

[1] Department of Software Engineering and Cybersecurity, Faculty of Computer Science, West Pomeranian University of Technology

[2] Department of Sustainable Finance and Capital Markets, Institute of Economics and Finance, University of Szczecin, 70-453 Szczecin, Poland

* Correspondence: wwysocki@wi.zut.edu.pl

Abstract: Managing software development processes is still a serious challenge and offers the possibility of introducing improvements that will reduce the resources needed to complete projects successfully. The article presents the original concept of classification of types of project tasks, which will allow for more beneficial use of the collected data in management support systems in the IT industry. The currently used agile management methods, described in the article, and the fact that changes in the course of projects are inevitable were the inspiration for creating sets of tasks that occur in software development. Thanks to statistics for generating tasks and aggregating results in an iterative and incremental way, the analysis is more accurate and allows you to plan the further course of work in the project, select the optimal number of employees in task teams and identify bottlenecks that may decide on a faster completion of the project with success. The use of data from actual software projects in the IT industry made it possible to classify the types of tasks and the necessary values for further work planning depending on the nature of the planned software development project.

Keywords: software; knowledge management; reasoning; information extraction; rules mining; knowledge acquisition and engineering

---

1. Introduction

In the modern era of information society, computers are a basic work tool. Tasks performed by information technology can be simple for today, i.e. not requiring high computing power, e.g. document processing, or they can be characterized by high computational complexity, e.g. services provided by data centers. Such a variety of tasks means that many types of IT equipment and computers dedicated to specific applications are available. For example, the market offers ultrabooks, i.e. portable computers with low energy consumption, laptops (general purpose portable computers), or typical stationary computers (desktops) to efficient workstations, both portable and stationary. However, all IT devices have a common feature from an electrical point of view, namely they are nonlinear receivers with a variable load value. This nonlinearity is related to the presence of rectifier systems in power supplies. On the other hand, load variability is related to the way of work, where it is characteristic that complex computational tasks require increased power consumption. Therefore, the right selection of computer equipment, not only in terms of performance, but also electricity consumption can contribute to reducing electricity consumption, which directly affects the operating costs of the management organization. In addition, current computers are equipped with advanced technical solutions that help optimize (minimize) energy consumption, which in the case of data centers and an increasing percentage of IT devices used is becoming extremely important not only in financial terms, but also for the environment and sustainable development of the global economy.

Software is an integral part of working life, and it is used in various industries and companies to manage processes, communication, and data analysis. During our daily commute, we use navigation apps to inform us of current traffic and optimise our route. Childcare providers use apps to organise schedules, monitor safety of children and to communicate with their parents. In the kitchen, software makes it easier to plan meals, access recipes, and control the cooking process with special applications. In government offices, software enables electronic errands, reducing waiting times and facilitating access to public services. In healthcare and medicine software supports diagnosis, treatment, and patient data management, thus improving overall quality of healthcare. Most of these applications require interaction with services and components on the global IT network. Software is ubiquitous and has thus enabled a giant leap in civilisation. Reducing its use or abandoning digital technologies would result in an inevitable regression. However, software requires computers and communication links, which means even more hardware. That hardware, in turn, consumes huge amounts of energy.

Already, a study conducted by the Boston Consulting Group has shown that the Internet is responsible for 2.4% of total global greenhouse gas emissions, which is equivalent to the entire aviation industry [1]. Undoubtedly, electricity is one of the most essential media for life. However, three years ago, in 2020, many organizations and modern enterprises did not deal with the issue of measuring energy consumption in as many aspects as they do now. Although, of course, this parameter was in the consciousness, and its values have always been important. However, with the passage of time, the power consumption factor has become much more important, including primarily the price, but also the fundamental impact on the environment and the global economy has been noticed. This has caused an unprecedented need to look for ways to save this resource and reduce production costs. The information society, having measured energy consumption, allows its use to be optimized in many ways. The use of a meter for specific devices is currently necessary due to the possibility of using many methods of optimizing energy consumption. Of course, this also allows for drawing more precise conclusions and applying appropriate methods for reducing or including the costs incurred in the appropriate prices of products or services, as is the case with data centers. The great interest in resources, consumption measurements and methods for optimizing the use of electricity is undoubtedly influenced by the drastically rising prices and the difficult to predict large fluctuations in their changes.

At the moment, there is no way to slow down technological development without a dramatic drop in the quality of life. Therefore, we need to consider how to best use existing resources so that future generations can live in a world no worse than the one we live in. The article is divided into several chapters. The first chapter shows where the impulse to start work came from. The second chapter presents the methodology of the conducted research, including the research questions. The third chapter shows the results of a literature review oriented according to the order of decisions made in the software development processes. Its aim is to support decision-makers in searching for energy-efficient solutions. The fourth chapter presents the current state of application of good practices and methods for reducing energy consumption by software. The fifth chapter summarizes the research results, presents conclusions and shows directions for changing the current situation. The aim of the article is to show the need to optimize the use of electricity by computer devices in accordance with the concept of sustainable development and to analyze methods of reducing energy consumption. The article discusses the most popular techniques for reducing energy consumption by software, which is of fundamental importance for this type of organization and translates into the costs of services provided, and indirectly affects the future of the global economy.

2. Motivation

At the end of 2021, the English edition of Le Monde Diplomatique published an article by Guilliam Pitron, "No Such Place as the Cloud" [2]. The article broadly discusses the impact of digital technologies on the natural environment. The author, as a journalist who has been dealing with the geopolitics of rare earth metal extraction for many years [3], presents this time not only the issues of the direct impact of the increasing extraction of raw materials in order to construct the latest digital equipment. He also discusses the negative consequences of software on the environment of our planet.

Figure 1 shows a map of the problems described in the article. Thanks to the new method of estimating environmental costs MIPS, it turned out that the more advanced the production technology, the higher the environmental costs [4]. This applies to the most advanced technologies of electronic equipment production. This equipment is used to build the largest infrastructure on our planet. The article's author cited an estimate that in 2022 the data centers that comprise it will consume about 10% of global energy production. All this growing infrastructure is used to store and process data. Software is largely responsible for energy consumption. Reading the article was a surprise for the authors of the article, people with extensive experience in the ICT industry (programming, IT project management and education). The image seen from a great height looks different than from the perspective of a person working in the industry. Many employees still live in the myth that software is something abstract, intangible and software production is creating something out of nothing. It is difficult to understand how these intangible entities can threaten the natural environment or limit the ability of future generations to meet basic needs. However, pollution generated by digital technology is a fact. After recognizing the importance of the problem, the authors decided to look for solutions to improve the situation in their own areas of competence.

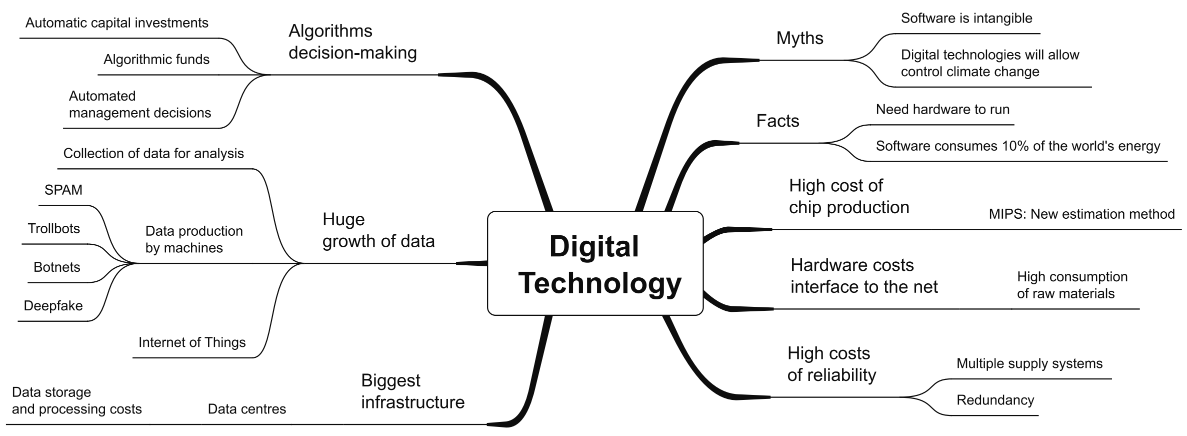
Digital Technology
Algorithms decision-making
Automatic capital investments
Algorithmic funds
Automated management decisions
Huge growth of data
Collection of data for analysis
Data production by machines
SPAM
Trollbots
Botnets
Deepfake
Internet of Things
Biggest infrastructure
Data centres
Data storage and processing costs
Myths
Software is intangible
Digital technologies will allow control climate change
Facts
Need hardware to run
Software consumes 10% of the world's energy
High cost of chip production
MIPS: New estimation method
Hardware costs interface to the net
High consumption of raw materials
High costs of reliability
Multiple supply systems
Redundancy

Figure 1. Map of problems caused by digital technology.

3. Research Method

In this article, we followed the guidelines for conducting a systematic literature review [5] with the consideration of process changes for a small research team and guidelines for conducting a light review. We performed the following steps of the procedure for systematically searching and selecting relevant sources:

Step 1. Define the research objective and research questions.

The aim of the research is to answer the question of how software engineering can contribute to the sustainable development of digital technologies. The literature review is to answer the following research questions:

RQ1: What are the possibilities (methods, practices, patterns) for reducing the energy consumption of software?

RQ2: What is the current state of application of existing methods and practices in the software industry?

RQ3: How can existing solutions be initiated in the industry?

Step 2. Conduct a pilot search.

After defining the research objective and research questions, we conducted pilot searches using the Google Scholar search engine and various queries. The results of the initial searches allowed us to orient ourselves in the existing literature and appropriately select the words in the search string.

Step 3. Define the search string.

The query was intended to find articles on software energy consumption or software energy efficiency. The phrase green software directly indicates the topic of interest. Hence, the query consists of six elements connected by logical connectives. As a result, we constructed the following query:

"((((energy OR power) AND (efficiency OR consumption)) OR green) AND software)".

In addition to the results of the main query, we included words that specified the topic of the searched studies, e.g. technique, practice, methodology, requirement, architecture, algorithm, approximate, survey, etc.

Step 4. Specify the searched article databases.

First, we used the Google Scholar search engine because it is easy to use, provides a wide range of materials in the search, and thus provides a broad perspective on the problem and its solutions. Second, we used the IEEE Xplore [6], ScienceDirect [7], and Springer Link [8] search engines. In order to expand the article database, we also used the snowballing method [9].

Step 5. Defining the inclusion and exclusion criteria

We were interested in articles from journals and conferences on software engineering in English. We searched for texts that answered our research questions, i.e. contained descriptions of methods, methods, techniques, practices, preferably verified in practice. We were also interested in articles, surveys, reviews showing the state of application of these methods in practice and proposals for improving the situation. The articles had to be publicly available. As a result, we received a set of inclusion and exclusion criteria, which we present in Table 1.

Table 1. Inclusion and exclusion criteria.

Criteria	**Inclusion**	**Exclusion**
Type of Publication	Journal, Conference Paper, Review,	
Full Text	Available	Not Available
Language	English	Other languages
Topic	Software Engineering practices, methods, approaches and general purpose algorithms	Out of Software Engineering, Limited to Mobile, Networking and Other Specific Applications
Models	Software Energy Consumption Estimation Models	Power consumption of ML and DL models

Step 6. Conducting the main search

We conducted the main search in December 2022. As a result, Google Scholar returned about 1,870,000 results. The results were sorted in order of best matching.

Step 7. Selection of publications by title, keywords and abstract

Then, publications were reviewed based on title, abstract and keywords. At this stage, publications on topics other than software engineering, or dealing with energy consumption by ML and DL models, as well as publications without full text available, were eliminated. The results were reviewed in pages containing 20 results. The number of matching articles decreased on subsequent pages, so the search was stopped after page 8.

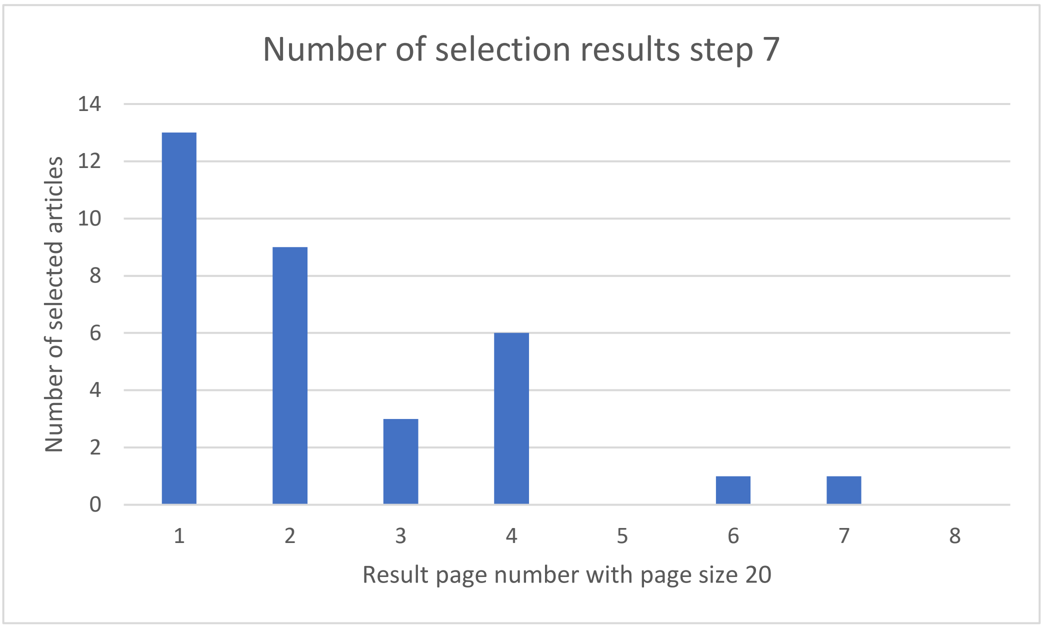

Figure 2. Number of articles matching criteria in Google Scholar search results pages.

In this way, we reviewed 8 pages of results, or 160 articles, and found 33 publications of interest to us.

Step 8. Read the full text:

We read the remaining 33 in full. We received a list of seven articles suitable for inclusion in the review. This is of course far too few to review software energy efficiency methods, examine the situation in the industry, and draw conclusions on how to improve it. Therefore, we decided to use an agile approach and dynamically changed the query.

Another argument for moving away from a rigid protocol is that systematic reviews were created to collect medical research results and then build large data sets from them that would allow for meta-studies. Our goal was different, so we used the IEEE Xplore [6], ScienceDirect [7], and Springer Link [8] search engines to create a lightweight review. We attached words specifying the topic of the searched studies to the single software engineering disciplines in the order shown in Figure 3. For example, these were: technique, practice, methodology, requirement, architecture, algorithm, approximate, survey, etc. To retrieve more articles, we used the snowballing method [10]. This resulted in 98 articles, 68 of which we used in the article.

4. Literature Review

A systematic review of articles in the field of software engineering on the energy consumption of IT systems and applications revealed a wide range of methods, best practice techniques that address the problem and have been validated by researchers and practitioners in the IT industry. This provided an answer to the first research question

RQ1: What are the possibilities to reduce the energy consumption of software?

In this article, we provide an overview of research articles that consider different aspects of software engineering and their impact on energy consumption. Figure 3 shows them in the order discussed in the article.

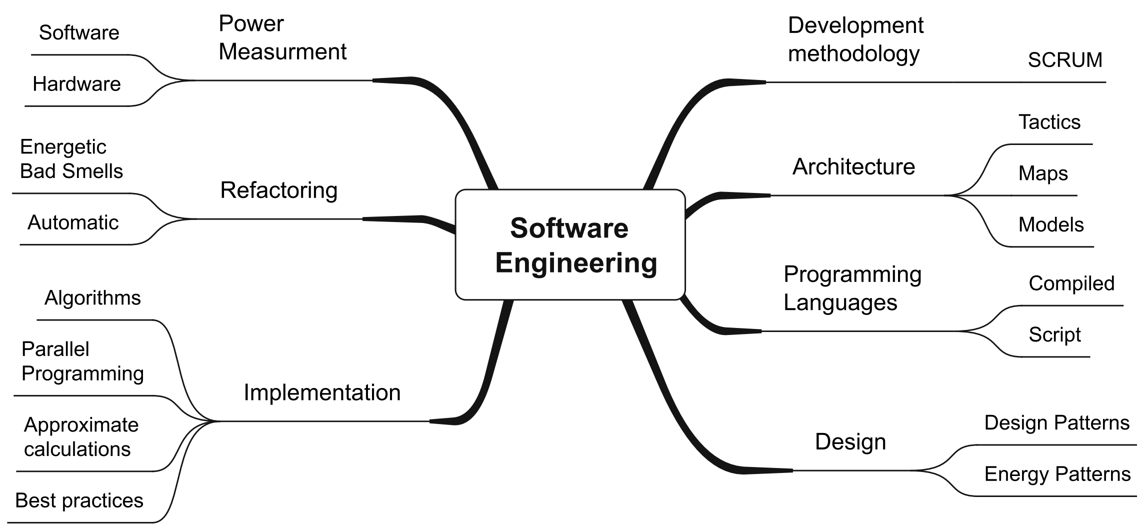

Figure 3. Software engineering aspects affecting energy consumption.

Software engineering is the application of a systematic, disciplined and quantifiable approach to the development, testing, implementation, operation and maintenance of software systems [11]. There are various approaches to dividing the above disciplines into processes and sub-processes performed sequentially or in parallel. These approaches are called software life cycle methodologies or models. Each of the processes in these models is supported by recommended, proven practices and techniques. Are there any software development methodologies that are more supportive of energy-efficient software development? The known literature does not consider the answer at this level of generality. Currently, the most commonly used approaches to software development are agile approaches. As a result of their popularity, a greater number of projects end in success. Since we care about the smallest possible burden on the environment by software, we can say that agile methodologies are the most effective, because other approaches more often end in wasting time, effort, energy and money. The question of which of the agile practices most often support the development of efficient software is answered by the authors of a broad review of articles from 2005-2017 [12]. The results of the literature review were verified by means of a survey conducted among experts from the IT industry. The paper identifies 6 main success factors, which are supported by 36 identified practices. The results show that agile software development methods are emerging as the best way to produce green and sustainable software. A similar opinion is expressed by Dick et al., who developed a model that integrates Green IT aspects with software engineering processes using agile methods to create "greener" software from scratch [13]. Therefore, in this paper we consider software engineering from the point of view of agile methodologies.

4.1. Software Architecture

Designing the architecture of an IT system is the most important stage of design in traditional software development methodologies. Its purpose is to divide the system into components, determine how they work together, and indicate the technology. As a result of working on the architecture of the system being developed, we obtain greater coherence of components and less coupling between them. The role of the architect does not exist in scrum teams, the entire team is responsible for making such decisions. Decisions regarding architecture can be made at the very beginning of the development process, but in agile methodologies, these decisions are made after the software implementation begins. They make up the evolving architecture. This architecture consists of emergent decisions, from which the target architecture emerges, and refactoring decisions that change previous decisions. Regardless of when these decisions are made in the software development process, they have a great impact on the energy consumption of the software being created. The system architecture design process requires making a number of key decisions made on the basis of compromise. Decisions concern how to meet non-functional requirements and key functional requirements of the customer. In this way, it is strongly connected to the area of requirements acquisition. In agile methodologies, we do not know all the requirements at the beginning; we learn the details during the project work. For green IT systems, it is important to have clearly defined, verifiable requirements for the energy efficiency of the software. Seyff et al. describe a Win-Win model of requirements negotiation extended with sustainability [12]. On this basis, appropriate decisions can be made and verified in practice. Solid support for architectural decisions based on a model that allows forecasting energy consumption with an error below 6% is presented in the article [14]. Architectural tactics, i.e. decisions based on quality measures, are proposed by the authors of the article [15]. Thanks to the perspective of energy consumption by the software architecture, it was possible to reduce energy consumption by 67% in the example application. Support for making architectural decisions in the context of energy consumption is also provided by decision maps presented in the article [16]. Maintainability is an important quality in making architectural decisions. Its relationship with energy consumption by software is the subject of promising preliminary work described in the publication [17]. Fontana de Nardin et al. developed the Elergy model, which uses the advantages of the elasticity-driven approach of microservice architecture in cloud applications [18]. It allows to predict the load change of individual microservices and make a decision about scaling the service in advance. Its energy efficiency allows to save up to 27.92% of energy. The methods supporting decision-making at the architectural level and their effectiveness in reducing energy consumption are given in Table 1.

Table 1. Methods supporting the design of energy-efficient architecture.

Method	Reducing energy consumption
Negotiating Win-Win Sustainability Requirements [12]	-
Energy Consumption Model [14]	-
Architectural Tactics [15]	67%
Decision Maps [16]	-
Energy Model [18]	1.93% - 27.92%

At a similar level of generality are decisions about the choice of programming language and software technology. Below we present an overview of the information on this topic.

4.2. Programming Languages

A programming language allows us to express the collected knowledge about the system being built in the form of code. Decisions regarding the selection of the software language and related technologies, libraries, and frameworks are made at the beginning of the agile development process. They are related to the technological competences of the developers who make up the project teams. If we are building a monolithic system, it will be difficult to change the decision once made. In the case of choosing a microservice architecture, it is possible to delay decision-making until the emergence of subsequent business areas. Such an architecture makes it easier to build a system from components created in different technologies. Both the programming language and related technologies have a strong relationship with the energy consumption of the built software. In an organization producing green software, decisions regarding the selection of a programming language and technology should be made based on their energy efficiency. The energy consumption of different software languages has been the subject of many studies in recent years and is described in the following articles. Georgoiu et al. conducted an experiment comparing the energy consumption of nine basic tasks from the Rosetta Code collection [10] implemented in seven interpreted and compiled programming languages [19]. For simple tasks, the lowest energy consumption was measured for compiled languages with optimization enabled. For URL encoding and decoding, Java achieved the highest energy efficiency. Among interpreted languages, Javascript achieved the highest efficiency.

Pereira et al. compared the energy consumption, execution time, and memory usage of ten programming problems from the Computer Language Benchmark Game (CLBG) collection [20]. Measuring the execution of programs in 27 tested programming languages on a desktop machine with an Intel Core i5-4460 processor showed that compiled programming languages are usually the fastest and most energy efficient (C is the best for almost all tested problems) [21]. In second place in these respects are languages run in virtual machines (Java consumes twice as much energy as C). Interpreted languages turned out to be the least energy efficient and slowest (Python consumes 76 times more energy than C). The results of the research were later validated by the authors using tasks from the aforementioned Rosetta Code repository [22].

Oliveira et al. compared the energy consumption of a set of applications and benchmarks from Rosetta Code, CLBG, and F-Droid [23] on the Android mobile platform [24]. Their results indicate that for intensive computations, JavaSript is more energy efficient than Java. The authors suggest using a hybrid approach, running Java applications with JavaScript scripts to save energy. The summary of the energy consumption results of popular programming languages is given in Table 2.

Table 2. Comparison of energy consumption of different programming languages.

Comparison of Power Consumption by Programming Languages	Hardware Platform
Comparison of 6 languages, Rosetta Code tasks [19]	Laptop i Raspberry Pi
Comparison of 27 languages, CLBG tasks [20]	Desktop
Comparison of 27 languages, CLBG tasks and Rosetta Code [21]	Desktop
Comparison of Java and JavaScript [22]	Android

The entire team is responsible for software design in agile methodologies. One of the main decisions in the design of software components is the use of design patterns. Design patterns are proven solutions to problems that often appear in a specific context. The use of patterns allows for better understanding and modifying the code of the created component and increasing the flexibility of the solution. Feitosa et al. [1] describe both modern patterns oriented towards increasing energy efficiency and energy consequences of using standard design patterns.

Energy patterns understood as good practices reducing the energy consumption of mobile applications were described by Cruz et al. in the article [25]. It contains a catalog of 22 practices obtained as a result of searching in GitHub code repositories and then thematic analysis [26] and classification.

Schaarschmidt et al. built a framework for describing design patterns for energy-efficient embedded systems [27]. They proposed a numerical efficiency coefficient to calculate energy savings due to the use of the pattern. The framework was used to describe new and well-known design patterns.

Table 3 summarizes the research on the impact of design patterns on energy consumption. The next subsection presents the results of the search for works in the field of software implementation.

Table 3. Design Patterns as Support for Energy-Efficient Software Design.

Design Patterns	Hardware Platform
Modern and standard [1]	-
Energy [25]	Mobile
Framework for describing patterns [27]	Embedded

4.3. Software Implementation

In agile methodologies, software development is done in iterations lasting several weeks. At the beginning of each iteration, the team details the requirements and chooses which ones to implement. Then, they design, code, and test the emerging software functions. As a result of the iteration, the client receives another version of working software. Based on the feedback, the team adjusts the software implementation to the client's needs. In the part describing the implementation, we present research on reducing the energy consumption of lower-level elements, such as: selecting appropriate algorithms, the impact of parallel programming, using approximate calculations, and using good practices.

One way to increase the energy efficiency of software is to choose an appropriate algorithm that consumes less energy. Energy consumption by depends on the execution time, which results from the computational complexity of the algorithm. However, it does not exhaust all the causes. Among other things, locality of calculations is important, which increases the use of cache memory, performance, and energy efficiency.

There are comparisons of different versions of algorithms on different hardware platforms, which allow to choose the optimal energy solution. Rashid, Ardito & Torchiano performed an experiment, in which they measured the energy consumption of different sorting algorithms implemented in different programming languages on the Raspberry Pi device [28]. The results show which sorting algorithms implemented in which language consume the least energy. Schmitt et al. performed similar experiments comparing the energy efficiency of sorting algorithms on desktop computers [29].

Collections are abstract classes, which are used to store objects and allow to treat them as a single set of data. They allow to perform operations on it, e.g. adding, removing and browsing elements of the set. There are many types of collections, which differ in properties and implementation. The proper selection of a specific collection for the problem and context has an impact on increasing the performance and reducing the energy consumption of the application.

Pereira et al. measured the energy consumption of standard Java collections and used the measurement results to optimize the energy consumption of a set of sample applications [30]. In preliminary results, they achieved a 6.2% reduction in energy consumption. Hasan et al. created detailed profiles of the energy consumed by standard operations performed on Java abstract collections [31]. The results of the study show that using the wrong collection, according to the profiles, can result in a 300% increase in energy consumption. Optimizing the sample application using WALA analysis resulted in a 38% reduction in energy consumption and a speedup of about 1.6 times. Pinto et al. empirically studied 16 implementations of Java collections and conducted energy consumption measurements [32]. They obtained significant results in comparisons of individual implementations, where a 2.19 times reduction in energy consumption was achieved. For real Tomcat and Xalan applications, an improvement of 17% was obtained.

As we can see, by using appropriate types and implementations of collections, a significant reduction in energy consumption can be achieved. Another important group of algorithms are cryptographic algorithms, because they require a lot of computational power and therefore consume a significant amount of energy.

An important class of frequently used algorithms with high computational complexity are cryptographic algorithms. Classical algorithms used until recently are threatened on the one hand by the growing power of conventional computers and on the other hand by the ever closer perspective of introducing quantum computers with enormous computational power. Therefore, we tend to extend hashes and cryptographic keys and introduce new, more complex algorithms. In 2016, the American National Institute of Standards and Technology (NIST) began the process of standardizing post-quantum cryptography (PQC) algorithms [33]. The energy consumption of these algorithms is particularly important for mobile devices or IoT. C. A. Roma et al. conducted research on the energy efficiency of PQC algorithms [34] running on the Intel Core i7 processor. The obtained results for different algorithms and security levels allow to select the appropriate algorithm for the context.

Elsadeka et al. investigated the energy efficiency of lightweight cryptographic algorithms standardized by NIST [35]. Thanks to parallel processing, up to 49% and 28% increase in energy efficiency was achieved, and up to 96% and 45% for the Elephant and ISAP algorithms. Specialized processors are developed for IoT applications to reduce energy consumption. An example of such a solution is an energy-efficient cryptoprocessor supporting the post-quantum version of the TLS protocol [36]. This is a hybrid solution containing a low-power RISC-V microprocessor core, on which the software part is implemented, and hardware accelerators for AES, ECC and SHA2 encryption. Thanks to the use of accelerators, the energy consumption spent on encryption was reduced by 2 to 8 times. Even better energy efficiency results were obtained thanks to the hardware implementation of the lightweight cryptography algorithm PRESENT [37] for IoT devices. The authors report an increase in energy efficiency by 63 times compared to the standard AES algorithm.

The most energy-efficient cryptographic algorithms were implemented in hardware for use in IoT devices. Maybe it's time to consider hardware support for cryptography in desktop devices and server solutions for data centers? Table 4 summarizes the results of work on energy-efficient algorithms.

Table 4. Algorithms supporting energy-efficient software.

Algorithm	Hardware Platform	Reduction of energy consumption
Sorting		
Sorting algorithms, multiple languages [28]	Raspberry Pi	comparison
Sorting algorithms, multiple languages [29]	Desktop	comparison
Collections		
Standard [30]	Java	6%
WALA analysis [31]]	Java	38%
Power consumption measurements and matching [32]	Java	17%
Cryptography		
Energy efficiency of PQC algorithms [34]	Desktop	comparison
Elephant [35]	IoT	49%
ISAP [35]	IoT	28%
TLS protocol [36]	IoT	50% - 87,5%
PRESENT [37]	IoT	98,4%

4.4. Parallel Programming

Parallel programming is the process of dividing large problems into smaller ones and solving them in parallel. The interest in parallel programming is largely due to the physical limitations of increasing the frequency of processors. For this reason, modern processors consist of many cores enabling parallel operation.

A comprehensive review of software methods for improving the energy efficiency of parallel processing includes the most advanced methods for supercomputers intended for scientific computing [38]. The 2017 article presents energy efficiency as a major problem in the design of modern computing systems from supercomputers to laptops with multicore processors. The theoretical models described in the article show that energy efficiency is dependent on both the application and the platform; energy consumption and performance are closely correlated, and the choice of processor frequency has a smaller impact on efficiency than changing parallelism. The article presents methods for improving energy efficiency in five groups: resource management, parallelization optimization, communication, automatic energy tuning, and approximation. Although advanced methods are mainly related to scientific computations performed on supercomputers, they can be an inspiration for software solutions for data centers.

Kambadur and Kim worked on discovering configurations and parameters that will allow to reduce the energy consumption of parallel applications [39]. The authors measured the energy consumption and execution time of 41 test applications in 220 configurations. The results indicate that the most important factors for low energy consumption are the efficiency of parallelization and optimizations introduced by the compiler.

The literature review [40] also lists two general algorithms for minimizing energy consumption in multithreaded processing. Both reduce energy consumption by effectively managing threads. In the first one, DVFS (Dynamic Voltage Frequency Scaling) adjusts the supply voltage and changes the processor clock frequency depending on the current load. However, this approach does not work well when using multithreaded processors. To overcome this limitation, the authors use the thread shuffling technique, which allowed to reduce energy consumption by 56% [41]. In the HERMES solution, also based on DFVS, a strategy based on the thief-victim approach [42] is used. It consists of two algorithms, workpath-sensitive and workload-sensitive, which adjust the clock frequency of the processor core to its load. As a result of using the HERMES approach, the energy consumption can be reduced by about 3-4%. Table 5 summarizes the research results on the impact of parallel implementation on energy efficiency. The next subsection lists solutions based on adjusting the accuracy of calculations.

Table 5. Methods for Reducing Energy Consumption with Parallel Programming.

Method	Hardware Platform	Reduction of energy consumption
Methods overview [38]	Supercomputer	different
Configurations and parameters [39]	Desktop	-
Thread shuffling [41]	Supercomputer	56%
HERMES [42]	Supercomputer	3% - 4%

Approximate computing is an approach to reducing the precision of computations in order to save energy. In this approach, we can distinguish techniques based on approximate data types and on approximate iterations and functions. When using approximate data types, the most important problem is to separate data that should be calculated precisely from those that can be approximated.

Sampson et al. proposed the EnerJ language, which is an extension of the Java language introducing approximate data types [43]. Using EnerJ annotations, one can mark approximate and precise application data. Thanks to the annotations, the data is placed in approximate memory, and the code processing it has reduced energy consumption with reduced accuracy. The authors' experiments have shown that it is possible to reduce energy consumption by 10-50% with a small loss of accuracy.

Programmers often use double-precision floating-point variables. In many applications, such precision is not needed. Compared to single-precision variables, they occupy twice as much memory and twice the bandwidth. Dongarra et al. have evaluated the performance and energy consumption of PLASMA and FLAME numerical libraries using single precision variables and a hybrid approach [44]. The time and energy consumption were reduced by 50% for single precision and 25-30% for hybrid.

Loop perforation is a computational technique that changes the number of loops so that only a fraction of the iterations are executed, which reduces execution time and energy consumption. Hoffmann et al. have described a technique called code perforation that automatically extends the application to increase performance and energy efficiency at the expense of accuracy [45]. Applications of this technique include scientific computing, video and audio coding, Monte Carlo simulations, and machine learning algorithms. It is based on the SpeedPress compiler, which uses code perforation and the SpeedGuard module that continuously monitors the loss of precision and controls the code perforation accordingly.

A very effective approach to saving energy by simplifying computations is memoization. It consists of memorizing the results of a computationally intensive function for each parameter value. Agosta et al. proposed an approach based on dynamic memoization, which consists in automatic identification of functions to be memorized and automatic memorization of results for the most frequently used parameters [46]. They performed the verification for a set of financial functions. In practice, they achieved a reduction in energy consumption by 74% and an increase in efficiency by 79%. Approximate calculations enable a significant reduction in software energy consumption. Table 6 summarizes the mentioned studies on this topic.

Table 6. Methods for reducing energy consumption through approximation.

Method	Platform	Reduction of energy consumption
EnerJ [43]	Java	10% - 50%
Single precision, hybrid [44]	Supercomputer	50%, 25% - 30%
Memoization [46]	Java	74%

4.5. Good Practices

Best coding practices are sets of rules, formally or informally, established by different coding communities that help software practitioners improve software quality. A literature search found one reference on best coding practices for achieving energy-efficient programming. Pinto et al. searched StackOverflow, a popular developer resource for questions and answers about software energy efficiency [47]. The authors identified seven main causes of increased energy consumption cited by developers. These include unnecessary resource consumption, background activities, and excessive synchronization. They also identified eight solutions that were most frequently recommended on the resource. They then compared the recommended solutions with the state of the art and found numerous shortcomings. The solutions that are consistent with the state of the art are listed as best practices. Table 7 lists these practices. For details on the application of the practices and the contexts in which they are effective, we refer to the original article.

Table 7. Good practices for energy-efficient programming identified and described in [47].

Name of practice
Keep IO to a minimum
Bulk operations
Hardware Coordination
Concurrent programming
Race to Idle
Efficient Data structures
Loop transformations

Data compression
Offloading methods
Approximated programming

The remaining literature found refers to bad code smells, which are the result of not applying good practices. Refactoring is the essence of Agile methodologies. It is the process of improving the design of existing code without changing its behavior [48]. Thanks to refactoring, the source code becomes more readable, duplicates are eliminated, errors are easier to find and the complexity of the system is managed. Without refactoring, the software slowly degrades and it becomes increasingly difficult to make changes to it. Bad code smells are symptoms of deeper structural problems that should be recognized and refactored.

Cruz and Abreu described bad code smells associated with low energy efficiency of mobile applications [49]. In the next paper, they proposed a tool for automatic refactoring of source code that fixes five common problems in mobile applications [50].

Pinto et al. conducted an extensive review of the literature on energy refactorings published in 2015 [51]. The literature identifies 11 energy efficiency issues and presents the possibility of their refactoring. The following refactoring areas were identified: User Interfaces, CPU Offloading, HTTP Requests, I/O Operations, Continuously Running App, Excessive Copy Chains, Embrace Parallelism, GPU Programming, Approximate programming, Energy Types, Stream Programming. A detailed discussion of the areas can be found in the aforementioned publication.

Şanlıalp et al. conducted an energy efficiency analysis of five refactoring techniques and three of their combinations for 2048 desktop and 2048 mobile applications [52]. The energy consumption of applications written in C# and Java was measured using the Intel Power Gadget tool [53]. The analysis found that all refactoring techniques reduced energy consumption, but using a combination of incompatible techniques did not reduce energy consumption as much as expected. Table 8 presents a summary of energy efficiency refactoring methods.

Table 8. Refactoring Methods to Reduce Energy Consumption.

Method	Platform	Reduction of energy consumption
Automatic Refactoring [50]	Mobile	5%
Review of Problems and Refactoring [51]	Many	-
Automatic Refactoring [52]	Many	-

Work on reducing software energy consumption requires measuring the actual energy consumption. Thanks to feedback, we can assess the effectiveness of the solutions used and work on changes in the right direction. Measurements sometimes require measuring the total energy consumed by software and hardware, and sometimes measuring the energy consumption of a specific hardware component and the software component cooperating with it. Hardware methods, such as black-box methods, are best for measuring the total energy consumption. To optimize energy efficiency, we most often need information about the impact of the software component on energy consumption. In such cases, the best measurement methods are white-box methods, i.e. software methods.

The simplest solution used in the articles is to use the electrical engineering method to measure the total energy consumed by the device on which the software is running. Wattmeter - can be used to determine the energy consumption of the software in both mobile and embedded devices, as well as in computers and servers. Thanks to them, the standard electrical engineering method of power measurement can be used.

The studies described in the articles [28] and [29] use the NI USB-6210 [54] and Yokogawa WT210 power analyzer [55] devices. Another of these devices is Whatts Up? [56] - most often used to study the energy consumption of mobile devices. This is a power meter that saves up to 2000 measurements in the device's memory, which is read and presented in the application on a PC. The application communicates with the device via a USB connection. GreenMiner [57] is a device dedicated to testing the energy consumption of smartphones. It physically measures the energy consumption of mobile devices (Android phones) and automates application testing and reporting measurements to developers and researchers. There are also specialized devices, such as EET (Energy Efficiency Tester) [58]. It is a hardware energy efficiency meter. It is used to measure power and store measurement results for a PC running the analyzed software. It was designed as a component of the FEETINGS framework (Framework for Energy Efficiency Testing to Improve eNviromental Goals of the Software). The framework supports automatic collection of energy consumption data from the most important components of the computer, along with processing and visualization of the collected data. Seflab (Software Energy Footprint Lab) [59] is a laboratory consisting of a server and a measuring computer. The server is equipped with component power sensors (processors, motherboard, memory, hard drive, fans). The measuring computer uses transducers to read the power consumption of the server components caused by the software running on it.

4.6. Software Tools

Software tools enable white-box measurements. This gives them a big advantage over hardware tools, as they enable measurement of energy consumption by components of the measured system, sometimes even with a resolution down to the methods or functions of the tested software. Measuring energy consumption before and after changes to a component or function allows for making rational decisions to increase the energy efficiency of the software being created. Profilers are specialized tools that allow for dynamic analysis of a program in terms of energy consumption. Software methods are mostly based on interfaces provided by modern processors. One of them is the Running Average Power Limit (RAPL) developed by Intel [60,61]. It is used to report the cumulative energy consumption of various domains: the CPU system, the attached DRAM memory, and the embedded graphics processor. Below is a list of tools described in the review articles or used in the described studies. These tools work on different hardware platforms and provide different measurement resolutions. AEP (Android Energy Profiler) [62] provides detailed information about the performance and energy consumption of any Android app in real time, without the need for source code or external power meters. Android Runner (AR) [63] is a framework for automatically executing measurement-based experiments in native and web applications. The main goal of AR is to streamline the execution of measurement-based experiments on Android devices. BitWatts [64] relies on distributed middleware to collect information about process utilization and infer detailed energy consumption without the need to invest in hardware (e.g. power meters).

IgProf [65] is an application profiler developed at CERN for scientific computing workloads. The energy profiler is based on sampling and obtains energy measurements from the Running Average Power Limit (RAPL) interface present in the latest Intel processors.

Jalen [66] is a software-level profiler. It is responsible for profiling running applications and estimating their energy consumption at a higher level, i.e. at the thread or method level.

Jolinar [67] is an open-source energy measurement tool that allows profiling a specific application at a given time. Jolinar can profile the energy consumed by various resources, such as CPU, RAM, and disk, except for the network card.

jRAPL [68] is an open source library that provides a set of APIs for profiling Java programs running on processors with Running Average Power Limit (RAPL) support.

PETRA (Power Estimation Tool for Android) [69] is able to measure the power consumption of Android applications based on the tools and APIs provided with the publicly available Project Volta 2. The tool provides an estimated measurement of power consumption at the method level.

PowerAPI [70] is an operating system level system monitoring library. It estimates the power consumption of running processes in real time, based on raw information collected from hardware devices (e.g. CPU, network card) via the operating system.

Power Gadget [53] is a software-based tool for monitoring the power consumption of Intel Core processors (2nd to 7th generation). It is supported on Windows and Mac OS X. It includes an application, driver, and libraries for monitoring and estimating real-time CPU power information in watts using in-processor energy meters. The product is no longer continued [71].

PowerJoular [72] enables monitoring of power consumption during operation of many hardware components of different devices and architectures. The initial version monitors the power consumption of CPUs and GPUs in computers and servers, and CPUs in Raspberry Pi devices.

Trepn Profiler [73] is a device power modeling and performance measurement tool to estimate the energy of the device, components, and smartphone applications.

A review of articles on methods and practices for reducing software power consumption also provides information on the measurement methods used. We have compiled a summary of the methods used in the described studies. The most commonly used method in the studies is the use of the RAPL interface [60,61]. The second most common is the use of various types of wattmeters. The remaining methods occur in isolated cases. Some of the methods described by us were not used in the studies. Table 9 shows the complete list of methods used in the studies described in the articles, supplemented with the results of the literature review, sorted by most frequently used and then alphabetically.

Table 9. A summary of the methods used in the research described in the articles, extended with the results of the literature review.

Method	Type	Platform	Research
RAPL [60]	software	windows, linux	[21,22,34,39]
PowerGadget [53]	software	windows, mac	[52,74]
Android Runner [63]	software	android	[75]
IgProf [65]	software	linux	[34]
jRAPL [68]	software	java	[30]
Trepn [73]	software	android	[76]
AEP [62]	software	android	-
BitWatts [64]	software	vms	-
GreenOracle [77]	software	android	-
Jolinar [67]	software	linux	-
PETRA [69]	software	android	-
PowerAPI [70]	software	linux	-
PowerJoular [72]	software	Linux, vms, gpu, raspberry PI	-
Power Meter	hardware	all	[46]
NI USB-6210 1 DAQ [54]	hardware	all	[28]
Yokogawa WT210 power analyzer [55]	hardware	all	[29]
Watts Up Pro [56]	hardware	all	[19]
EET [58]	hardware	all	[17]
GreenMiner [57]	hardware	all	[31]
Seflab [59]	hardware	server	-

4. Results

The problem of energy consumption in software has been noticed quite recently, both by the research community and by practitioners working on software development in the industry. In this article, we reviewed methods, techniques and practices that allow to reduce energy consumption in software. This is part of the achievements of the last ten years of research. In order to use these results in practice to create energy-efficient software, developers should be aware of both the existence of the problem and ready-made effective solutions.

In recent years, many authors have investigated this issue using surveys. The surveys included different groups of project team members. They concerned the awareness of the problem of energy consumption in software, the existence of methods to reduce consumption and supporting tools. The level of knowledge about methods of optimizing efficiency and measurement methods was also checked. The respondents also determined whether they used these methods and tools in practice. Additionally, one of the authors checked whether beliefs about climate change are correlated with awareness. Figure 4 shows a map of the aspects examined using surveys.

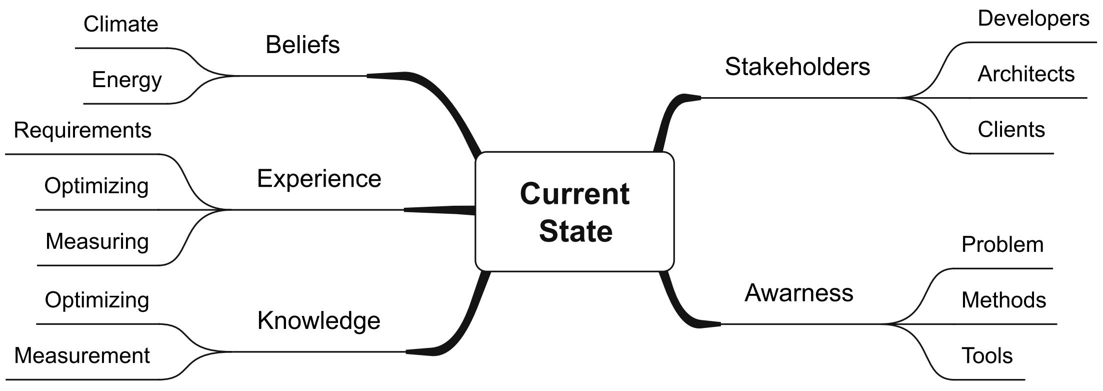

Figure 4. Aspects illustrating the current state of energy efficiency in the software industry.

By reviewing these studies, we were able to answer the second research question:

RQ2: What is the current state of application of existing methods and practices in the software industry?

Pang et al. conducted a survey among 122 software developers in 2013. The survey results, along with an analysis of the results in relation to the literature, were published in 2016 [78]. The anonymous online survey consisted of 13 questions, 10 of which concerned knowledge and experience with software energy consumption. Only 18% responded that they consider energy consumption when developing software. 14% considered reducing energy consumption as a non-functional requirement. 21% modified software to reduce energy consumption. To reduce energy consumption, one must first measure it. Only 10% of respondents responded that they did so. The analysis shows that developers have limited knowledge about energy efficiency, are not familiar with best practices to reduce software energy consumption, and are not sure how software consumes energy. These results emphasize the need for energy efficiency training.

Bashroush et al. described the issue of increasing energy consumption in data centers from the perspective of software architects [79]. They conducted a survey among 12 experienced software architects from different types of ICT organizations. When asked about energy consumption challenges in the past five years, only 17% of them answered positively that they had solved such problems. When asked whether they believed that energy consumption would become a major architectural problem in the next 5 years, 67% of them answered positively. When asked whether they had the right tools to solve energy efficiency problems at the design stage, only 25% agreed. The reasons for this state of affairs are the lack of knowledge about how design affects software energy efficiency, the lack of cooperation and information flow between organizational structures, and the lack of correlation between energy savings and the amount of payment for the service. The conclusions emphasize the need to establish a measure of software energy efficiency so that energy-efficient solutions can be compared and selected.

Pinto and Castor conducted a survey in 2017 among 62 developers who worked on applications for mobile platforms [80]. Over 68% of respondents were experienced developers with 8 or more years of experience. 60% of them experienced problems with energy consumption of the software they created. When asked if they had found the main cause of the problem, 50% of them did not answer. Among those who did answer, the recurring answers included background activities, GPS, and unnecessary resource usage. On the other hand, 32% of respondents did not observe a significant improvement in energy consumption after using their solutions. Of those who noticed an improvement, only 5% used specialist tools. The rest relied on subjective feelings and indicated websites as a source of information. Despite this, over 67% of them stated that the problem of energy consumption is important or very important. This shows that there is awareness of the energy consumption problem among developers of applications for mobile platforms. However, there is a lack of awareness of the existence of measurement methods and techniques and practices that solve the problems. There is also a lack of tools that would support energy measurement divided into hardware and software components and tools supporting code refactoring.

One of the authors of this article, W. Wysocki, conducted a telephone survey among employees of software development companies in early 2023 [81]. The main goal of the study was to explain why, despite the existence of many techniques, methods, and practices aimed at reducing software energy consumption, developers so rarely use them in the software industry. The results of the study show that the state of awareness among developers and customers still needs to be improved. Most things have remained unchanged since 2013, when Pang et al. [78] conducted their research. Developers, even if they were not familiar with the problem of software energy efficiency, were able to generate ideas for reducing and measuring energy consumption. The responses indicated flexibility and a willingness to solve problems rather than specific knowledge. We saw that few developers working in the software industry have experience in reducing software energy consumption. It was interesting to find that developers' belief that humans are not responsible for climate change correlates with their failure to consider the problem of software energy efficiency as important. Developers’ open statements showed that some developers believe that their customers should be the ones to demand reduced software energy consumption. Some developers believe that using cloud computing ensures energy efficiency of the software. A conversation with a data center employee revealed that economic factors (cost of data center services) do not force reduced software energy consumption.

Proposals to change the situation in the software industry and consequently in the world result from reflection on the current state. It is most often undertaken by authors conducting surveys. Figure 5 shows the main factors and their impact on improving the situation. The main problem emerging from the research is the too low awareness of developers and customers. Both groups are not fully aware of the existence of the problem and the possibilities of action. Developers do not know about existing solutions and tools, and customers do not know that their decisions affect the situation. The introduction of energy consumption standards for software may encourage institutional customers to order energy-efficient software. This in turn may launch training for managers and developers of the software industry, which will increase their knowledge and practical skills. The introduction of standard software efficiency labeling will make it easier for individual customers to make a decision to purchase an application. A separate path is to introduce the topic of ensuring software efficiency to the curriculum of higher education. Theoretical and practical training will introduce prepared, conscious staff for the industry to the market.

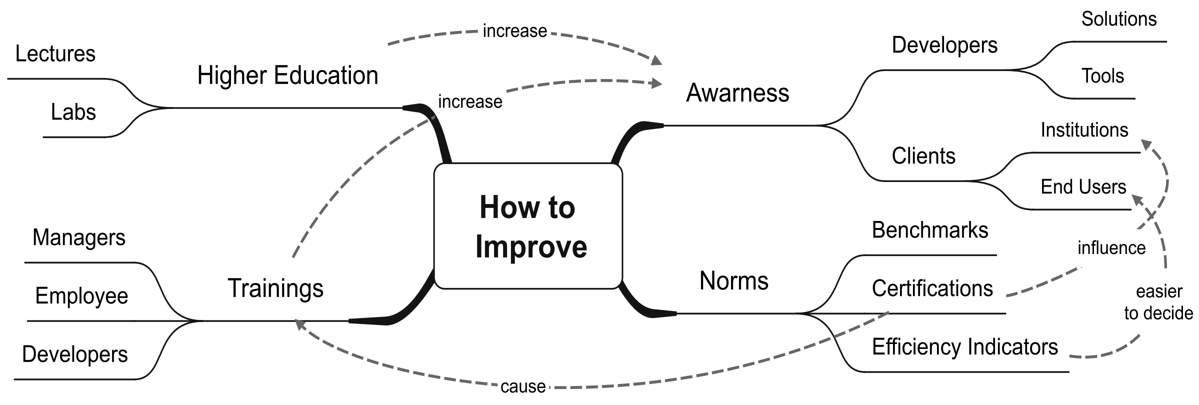

Figure 5. Opportunities for improvement and their synergies.

The described proposals for changes constitute an answer to the third research question:

RQ3: How can existing solutions be initiated in the industry?

Pang et al., based on the survey results, state that only 3 percent of respondents received complaints from users about excessive energy consumption [78]. According to them, this may indicate that users are not aware of the problem. Therefore, Zhang et al. propose the introduction of benchmarks, energy consumption standards and appropriate product labels [82]. Clearly defining the energy consumption class of applications can help increase users' awareness and help them make informed choices about the right product.

Pinto et al. [80] write that developers need to be more aware of the energy consumption problem. Currently, they do not know how to develop and maintain energy-efficient software systems. They also do not have sufficient support from tools.

Bashroush et al. remind that data centers are subject to standards such as the European Code of Conduct for Data Centres [83] and the Green Grid Data Centre Maturity Model [84]. Therefore, a similar software certification process seems inevitable.

Authors, based on the survey results [81], formulates the thesis that the most important thing is the awareness of the problem and existing ready-made solutions. Therefore, we should focus on the appropriate education of students who, when entering the job market, will change the software industry by becoming conscious developers, architects, managers and customers. The introduction of appropriate standards and certificates for software will launch professional training in the field of adapting products to standards, which will increase the awareness of both producers and institutional users.

5. Conclusions

Currently, the main problem emerging from the research is the low awareness of developers and customers. Both groups are not fully aware of the existence of the problem and the possibilities of action. Developers are not aware of existing methods and tools, while customers are not aware of the impact of their decisions. The introduction of energy consumption standards for software may encourage customers to order energy-efficient software. However, this situation will undoubtedly change due to the fact of rising prices and, at the same time, growing demand for electricity [85]. Because the desire to remain competitive on the market, which is a key issue for companies, will force intelligent energy management. Due to rising energy prices, there will undoubtedly be a huge interest in this topic. In fact, it has already happened, as proven by the systematic review of literature conducted in this article. Additionally, this trend will be strengthened by the active promotion of sustainable economic development and high quality of life. However, to achieve these goals, it will be necessary to effectively manage natural resources, and thus also optimize energy consumption, which is the foundation of a knowledge-based economy. At the same time, the development of the global economy is currently taking place particularly quickly in the areas of intelligent technologies, which forces an increasing demand for energy. And undoubtedly, interest in solutions of this type is growing dynamically due to the ongoing urbanization and digitization of more and more aspects of residents' lives. However, this also creates many new challenges, hence the need for intelligent solutions, including in the areas of public safety, urban mobility, or environmental monitoring, including, among others, the optimization of energy used to power all computer devices and new technologies.

Climate change is one of the most important challenges facing today's national economies. Extreme weather phenomena, rapidly rising river levels, and shrinking resources have an unprecedented impact on the functioning of entire economies. However, the existing mechanisms of functioning are unsustainable in the long term. Therefore, economies are focusing on more ecological goals, wanting to reduce the impact on the environment, and are implementing technologies for monitoring threats and the impact of extreme weather factors. In order to cope with the burden of climate change, it is necessary to optimize energy consumption. Additionally, with problems related to ensuring the security of energy supplies, it is necessary to take care of the optimization of energy use, which will give a fundamental advantage to such a modern economy.

More and more applications are making energy consumption critical. This is especially true for battery-powered devices and all mobile equipment. More and more suppliers of electronic components are paying great attention to these issues. This is of great importance for knowledge- and technology-based world economies. Because it has a direct impact on the operating costs of economic entities and thus on economic profitability. In addition, the use of energy optimization methods has a beneficial effect on the environment through the concept of sustainable development. Currently, there are many hardware and software methods for optimizing energy consumption, and we are only at the beginning of their search and wide-scale application. In many segments and from different sides, techniques for reducing the consumption of electricity, which is the foundation of every economy, should be analyzed and sought for the good of the environment and the possibility of sustainable development of the world. Modern developed countries are looking for solutions to this problem to enable the further development of world economies, which may be threatened by the insufficient amount of energy supplied to modern economies.

Author Contributions: Conceptualization, W.W. and I.M.; methodology, W.W.; software, W.W. and I.M.; validation, W.W. and I.M.; formal analysis, W.W. and I.M.; investigation W.W. and I.M.; resources, W.W. and I.M.; data curation, W.W. and I.M.; writing—original draft preparation, W.W. and I.M.; writing—review and editing, W.W. and I.M.; visualization, W.W. and I.M. ; supervision, W.W. and I.M.; project administration, W.W. and I.M.; funding acquisition, W.W. and I.M. All authors have read and agreed to the published version of the manuscript.

Funding: The project was financed within the framework of the program of the Minister of Science and Higher Education in Poland under the name "Regional Excellence Initiative" in the years 2024–2028.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Feitosa, D.; Cruz, L.; Abreu, R.; Fernandes, J.P.; Couto, M.; Saraiva, J.; Patterns and Energy Consumption: Design, Implementation, Studies, and Stories, in: C. Calero, M.Á. Moraga, M. Piattini (Eds.), Software Sustainability, Springer International Publishing, Cham, **2021**: pp. 89–121. https://doi.org/10.1007/978-3-030-69970-3_5.
2. No such place as the cloud, by Guillaume Pitron (Le Monde diplomatique - English edition, November 2021), (n.d.). https://mondediplo.com/2021/11/09digital-waste (accessed October 31, **2022**).
3. Pitron, G. The war for rare earth metals - The hidden face of the energy and digital transition, Editions Les Liens qui liberent, France, **2018**.
4. Saurat, M.; Ritthoff, M. Calculating MIPS 2.0, Resources 2, **2013**, 581–607. https://doi.org/10.3390/resources2040581.
5. Kitchenham, B. Procedures for Performing Systematic Reviews, Keele, UK, Keele Univ. 33, **2004**.
6. IEEE Xplore, (n.d.). https://ieeexplore.ieee.org/Xplore/home.jsp (accessed October 17, 2024).
7. ScienceDirect.com | Science, health and medical journals, full text articles and books., (n.d.). https://www.sciencedirect.com/ (accessed October 17, 2024).
8. Home | SpringerLink, (n.d.). https://link.springer.com/ (accessed October 17, **2024**).

9. Wohlin, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, Association for Computing Machinery, New York, NY, USA, **2014**: pp. 1–10. https://doi.org/10.1145/2601248.2601268.
10. Rosetta Code, Rosetta Code (2022). https://rosettacode.org/wiki/Rosetta_Code (accessed December 16, **2022**).
11. Jacobson, I.; Lawson, H.B.; Ng, P.-W. The Essentials of Modern Software Engineering: Free the Practices from the Method Prisons!, ACM Books, New York, NY, **2019**.
12. Seyff, N.; Betz, S.; Duboc, L.; Venters, C.; Becker, C.; Chitchyan, R.; Penzenstadler, B.; Nöbauer, M. Tailoring Requirements Negotiation to Sustainability, in: 2018 IEEE 26th International Requirements Engineering Conference (RE), **2018**: pp. 304–314. https://doi.org/10.1109/RE.2018.00038.
13. Dick, M.; Drangmeister, J.; Kern, E.; Naumann, S. Green software engineering with agile methods, in: 2013 2nd International Workshop on Green and Sustainable Software (GREENS), **2013**: pp. 78–85. https://doi.org/10.1109/GREENS.2013.6606425.
14. Stier, C.; Koziolek, A.; Groenda, H.; Reussner, R. Model-Based Energy Efficiency Analysis of Software Architectures, in: D. Weyns, R. Mirandola, I. Crnkovic (Eds.), Software Architecture, Springer International Publishing, Cham, **2015**: pp. 221–238. https://doi.org/10.1007/978-3-319-23727-5_18.
15. Jagroep, E.A.;. van der Werf, J.M.E.M.; Spauwen, R.; Blom, L.; van Vliet, R.; Brinkkemper, S. An Energy Consumption Perspective on Software Architecture, in: D. Weyns, R. Mirandola, I. Crnkovic (Eds.), Software Architecture, Springer International Publishing, Cham, **2015**: pp. 239–247. https://doi.org/10.1007/978-3-319-23727-5_19.
16. Lago, P. Architecture Design Decision Maps for Software Sustainability, in: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS), **2019**: pp. 61–64. https://doi.org/10.1109/ICSE-SEIS.2019.00015.
17. Mancebo, J.; Calero, C.; García, F. Does maintainability relate to the energy consumption of software? A case study, Software Qual J 29 (**2021**) 101–127. https://doi.org/10.1007/s11219-020-09536-9.
18. Fontana de Nardin, I.; da Rosa Righi, R.; Lima Lopes, T.R.; André da Costa, C.; Yeom, H.Y.; Köstler, H. On revisiting energy and performance in microservices applications: A cloud elasticity-driven approach, Parallel Computing 108 (**2021**) 102858. https://doi.org/10.1016/j.parco.2021.102858.
19. Georgiou, S.; Kechagia, M.; Spinellis, D. Analyzing Programming Languages' Energy Consumption: An Empirical Study, in: Proceedings of the 21st Pan-Hellenic Conference on Informatics, Association for Computing Machinery, New York, NY, USA, **2017**: pp. 1–6. https://doi.org/10.1145/3139367.3139418.
20. Zhang, Y.; Zhang, Y.; Portokalidis, G.; Xu, J. Towards Understanding the Runtime Performance of Rust, in: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, ACM, Rochester MI USA, **2022**: pp. 1–6. https://doi.org/10.1145/3551349.3559494.
21. Pereira, R.; Couto, M.; Ribeiro, F.; Rua, R.; Cunha, J.; Fernandes, J.P.; Saraiva, J. Energy efficiency across programming languages: how do energy, time, and memory relate?, in: Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering, Association for Computing Machinery, New York, NY, USA, **2017**: pp. 256–267. https://doi.org/10.1145/3136014.3136031.
22. 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 (**2021**) 102609. https://doi.org/10.1016/j.scico.2021.102609.
23. F-Droid - Free and Open Source Android App Repository, (n.d.). https://f-droid.org/ (accessed December 27, **2022**).
24. Oliveira, W.; Torres, W.; Castor, F.; Ximenes, B.H. Native or Web? A Preliminary Study on the Energy Consumption of Android Development Models, in: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), IEEE, Suita, **2016**: pp. 589–593. https://doi.org/10.1109/SANER.2016.93.
25. Cruz, L.; Abreu, R. Catalog of energy patterns for mobile applications, Empir Software Eng 24 (**2019**) 2209–2235. https://doi.org/10.1007/s10664-019-09682-0.
26. Fereday, J.; Muir-Cochrane, E. Demonstrating Rigor Using Thematic Analysis: A Hybrid Approach of Inductive and Deductive Coding and Theme Development, International Journal of Qualitative Methods 5 (**2006**) 80–92. https://doi.org/10.1177/160940690600500107.
27. Schaarschmidt, M.; Uelschen, M.; Pulvermüller, E.; Westerkamp, C.; Framework of Software Design Patterns for Energy-Aware Embedded Systems:, in: Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering, SCITEPRESS - Science and Technology Publications, Prague, Czech Republic, **2020**: pp. 62–73. https://doi.org/10.5220/0009351000620073.
28. Rashid, M.; Ardito, L.; Torchiano, M. Energy Consumption Analysis of Algorithms Implementations, in: 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE, Beijing, China, **2015**: pp. 1–4. https://doi.org/10.1109/ESEM.2015.7321198.
29. Schmitt, N.; Kamthania, S.; Rawtani, N.; Mendoza, L.; Lange, K.-D.; Kounev, S. Energy-Efficiency Comparison of Common Sorting Algorithms, in: 2021 29th International Symposium on Modeling,

Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), **2021**: pp. 1–8. https://doi.org/10.1109/MASCOTS53633.2021.9614299.
30. Pereira, R.; Couto, M.; Saraiva, J.; Cunha, J.; Fernandes, J.P. The influence of the Java collection framework on overall energy consumption, in: Proceedings of the 5th International Workshop on Green and Sustainable Software, Association for Computing Machinery, New York, NY, USA, **2016**: pp. 15–21. https://doi.org/10.1145/2896967.2896968.
31. Hasan, S.; King, Z.; Hafiz, M.; Sayagh, M.; Adams, B.; Hindle, A. Energy profiles of Java collections classes, in: Proceedings of the 38th International Conference on Software Engineering, Association for Computing Machinery, New York, NY, USA, **2016**: pp. 225–236. https://doi.org/10.1145/2884781.2884869.
32. Pinto, G.; Liu, K.; Castor, F.; Liu, Y.D. A Comprehensive Study on the Energy Efficiency of Java's Thread-Safe Collections, in: 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME), **2016**: pp. 20–31. https://doi.org/10.1109/ICSME.2016.34.
33. Moody, D.; Chen, L.; Jordan, S.; Liu, Y.-K.; Smith, D.; Perlner, R.; Peralta, R. NIST Report on Post-Quantum Cryptography, **2016**. https://doi.org/10.6028/NIST.IR.8105.
34. Roma, C.A.; Tai, C.-E.A.; Hasan, M.A. Energy Efficiency Analysis of Post-Quantum Cryptographic Algorithms, IEEE Access 9 (**2021**) 71295–71317. https://doi.org/10.1109/ACCESS.2021.3077843.
35. Elsadek, I.; Aftabjahani, S.; Gardner, D.; MacLean, E.; Wallrabenstein, J.R.; Tawfik, E.Y. Energy Efficiency Enhancement Of Parallelized Implementation of NIST Lightweight Cryptography Standardization Finalists, in: 2022 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE, Austin, TX, USA, **2022**: pp. 138–141. https://doi.org/10.1109/ISCAS48785.2022.9937755.
36. Banerjee, U.; Das, S.; Chandrakasan, A.P. Accelerating Post-Quantum Cryptography using an Energy-Efficient TLS Crypto-Processor, in: 2020 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE, Seville, Spain, **2020**: pp. 1–5. https://doi.org/10.1109/ISCAS45731.2020.9180550.
37. Goyal, T.K.; Sahula, V.; Kumawat, D. Energy Efficient Lightweight Cryptography Algorithms for IoT Devices, IETE Journal of Research 68 (**2022**) 1722–1735. https://doi.org/10.1080/03772063.2019.1670103.
38. Jin, C.; De Supinski, B.R.; Abramson, D.; Poxon, H.; DeRose, L.; Dinh, M.N.; Endrei, M.; Jessup, E.R. A survey on software methods to improve the energy efficiency of parallel computing, The International Journal of High Performance Computing Applications 31 (**2017**) 517–549. https://doi.org/10.1177/1094342016665471.
39. Kambadur, M.; Kim, M.A. An experimental survey of energy management across the stack, in: Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications, Association for Computing Machinery, New York, NY, USA, **2014**: pp. 329–344. https://doi.org/10.1145/2660193.2660196.
40. Georgiou, S.; Rizou, S.; Spinellis, D.; Software Development Lifecycle for Energy Efficiency: Techniques and Tools, ACM Comput. Surv. 52 (**2020**) 1–33. https://doi.org/10.1145/3337773.
41. Cai, Q.; Gonzalez, J.; Magklis, G.; Chaparro, P.; Gonzalez, A. Thread shuffling: Combining DVFS and thread migration to reduce energy consumptions for multi-core systems, in: IEEE/ACM International Symposium on Low Power Electronics and Design, IEEE, Fukuoka, Japan, **2011**: pp. 379–384. https://doi.org/10.1109/ISLPED.2011.5993670.
42. Ribic, H.; Liu, Y.D. Energy-efficient work-stealing language runtimes, in: Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems, Association for Computing Machinery, New York, NY, USA, **2014**: pp. 513–528. https://doi.org/10.1145/2541940.2541971.
43. Sampson, A.; Dietl, W.; Fortuna, E.; Gnanapragasam, D.; Ceze, L.; Grossman, D. EnerJ: approximate data types for safe and general low-power computation, in: Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation, Association for Computing Machinery, New York, NY, USA, **2011**: pp. 164–174. https://doi.org/10.1145/1993498.1993518.
44. Dongarra, J.; Ltaief, H.; Luszczek, P.; Weaver, V.M. Energy Footprint of Advanced Dense Numerical Linear Algebra Using Tile Algorithms on Multicore Architectures, in: 2012 Second International Conference on Cloud and Green Computing, **2012**: pp. 274–281. https://doi.org/10.1109/CGC.2012.113.
45. Agarwal, A.; Rinard, M.; Sidiroglou, S.; Misailovic, S.; Hoffmann, H. Using Code Perforation to Improve Performance, Reduce Energy Consumption, and Respond to Failures, (**2009**).
46. Agosta, G.; Bessi, M.; Capra, E.; Francalanci, C. Dynamic memoization for energy efficiency in financial applications, in: 2011 International Green Computing Conference and Workshops, **2011**: pp. 1–8. https://doi.org/10.1109/IGCC.2011.6008559.
47. Pinto, G.; Castor, F.; Liu, Y.D. Mining questions about software energy consumption, in: Proceedings of the 11th Working Conference on Mining Software Repositories, Association for Computing Machinery, New York, NY, USA, **2014**: pp. 22–31. https://doi.org/10.1145/2597073.2597110.
48. Fowler, M. Refactoring: improving the design of existing code, Second edition, Addison-Wesley, Boston, **2019**.

49. Cruz, L.; Abreu, R. Performance-Based Guidelines for Energy Efficient Mobile Applications, in: 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft), IEEE, Buenos Aires, Argentina, **2017**: pp. 46–57. https://doi.org/10.1109/MOBILESoft.2017.19.
50. Cruz, L.; Abreu, R. Improving Energy Efficiency Through Automatic Refactoring, JSERD 7 (**2019**) 2. https://doi.org/10.5753/jserd.2019.17.
51. Pinto, G.; Soares-Neto, F.; Castor, F. Refactoring for Energy Efficiency: A Reflection on the State of the Art, in: 2015 IEEE/ACM 4th International Workshop on Green and Sustainable Software, IEEE, Florence, **2015**: pp. 29–35. https://doi.org/10.1109/GREENS.2015.12.
52. Şanlıalp, I.; Öztürk, M.M.; Yiğit, T. Energy Efficiency Analysis of Code Refactoring Techniques for Green and Sustainable Software in Portable Devices, Electronics 11 (**2022**) 442. https://doi.org/10.3390/electronics11030442.
53. Intel® Power Gadget, Intel (n.d.). https://www.intel.com/content/www/us/en/developer/articles/tool/power-gadget.html (accessed December 23, **2022**).
54. USB-6210 - NI, (n.d.). https://www.ni.com/pl-pl/shop/model/usb-6210.html (accessed October 22, **2024**).
55. WT210/WT230 Digital Power Meters | Yokogawa Test&Measurement Corporation, (n.d.). https://tmi.yokogawa.com/eu/solutions/discontinued/wt210wt230-digital-power-meters/ (accessed October 22, **2024**).
56. Watts Up Pro Portable Power Meter, (n.d.). https://www.powermeterstore.com/p1206/watts_up_pro.php (accessed October 21, **2024**).
57. Hindle, A.; Wilson, A.; Rasmussen, K.; Barlow, E.J.; Campbell, J.C.; Romansky, S. GreenMiner: a hardware based mining software repositories software energy consumption framework, in: Proceedings of the 11th Working Conference on Mining Software Repositories, Association for Computing Machinery, New York, NY, USA, **2014**: pp. 12–21. https://doi.org/10.1145/2597073.2597097.
58. Mancebo, J.; Garcia, F.; Arriaga, H.; Moraga, M.; Guzmán, I.; Calero, C. EET: a device to support the measurement of software consumption, **2018**. https://doi.org/10.1145/3194078.3194081.
59. Ferreira, M.A.; Hoekstra, E.; Merkus, B.; Visser, B.; Visser, J. Seflab: A lab for measuring software energy footprints, in: 2013 2nd International Workshop on Green and Sustainable Software (GREENS), **2013**: pp. 30–37. https://doi.org/10.1109/GREENS.2013.6606419.
60. Hähnel, M.; Döbel, B.; Völp, M.; Härtig, H. Measuring energy consumption for short code paths using RAPL, SIGMETRICS Perform. Eval. Rev. 40 (**2012**) 13–17. https://doi.org/10.1145/2425248.2425252.
61. RAPL in Action, (n.d.). https://dl.acm.org/doi/epdf/10.1145/3177754 (accessed October 22, **2024**).
62. X. Chen, Z. Zong, Android App Energy Efficiency: The Impact of Language, Runtime, Compiler, and Implementation, in: 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom), **2016**: pp. 485–492. https://doi.org/10.1109/BDCloud-SocialCom-SustainCom.2016.77.
63. Malavolta, I.; Grua, E.M.; Lam, C.Y.; de Vries, R.; Tan, F.; Zielinski, E.; Peters, M.; Kaandorp, L. A Framework for the Automatic Execution of Measurement-based Experiments on Android Devices, in: 2020 35th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW), **2020**: pp. 61–66. https://doi.org/10.1145/3417113.3422184.
64. Colmant, M.; Kurpicz, M.; Felber, P.; Huertas, L.; Rouvoy, R.; Sobe, A. BitWatts: a process-level power monitoring middleware, in: Proceedings of the Posters and Demos Session of the 15th International Middleware Conference, Association for Computing Machinery, New York, NY, USA, **2014**: pp. 41–42. https://doi.org/10.1145/2678508.2678529.
65. Khan, K.; Nybäck, F.; Ou, Z.; Nurminen, J.; Niemi, T.; Eulisse, G.; Elmer, P.; Abdurachmanov, D. Energy Profiling Using IgProf, **2015**. https://doi.org/10.1109/CCGrid.2015.118.
66. Noureddine, A.; Rouvoy, R.; Seinturier, L. Monitoring energy hotspots in software, Automated Software Engg. 22 (**2015**) 291–332. https://doi.org/10.1007/s10515-014-0171-1.
67. Noureddine, A.; Islam, S.; Bashroush, R. Jolinar: analysing the energy footprint of software applications (demo), in: Proceedings of the 25th International Symposium on Software Testing and Analysis, Association for Computing Machinery, New York, NY, USA, **2016**: pp. 445–448. https://doi.org/10.1145/2931037.2948706.
68. Liu, K.; Pinto, G.; Liu, Y.D. Data-Oriented Characterization of Application-Level Energy Optimization, in: A. Egyed, I. Schaefer (Eds.), Fundamental Approaches to Software Engineering, Springer, Berlin, Heidelberg, **2015**: pp. 316–331. https://doi.org/10.1007/978-3-662-46675-9_21.
69. Di Nucci, D.; Palomba, F.; Prota, A.; Panichella, A.; Zaidman, A.; De Lucia, A. PETrA: A Software-Based Tool for Estimating the Energy Profile of Android Applications, in: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), **2017**: pp. 3–6. https://doi.org/10.1109/ICSE-C.2017.18.
70. Bourdon, A.; Noureddine, A.; Rouvoy, R.; Seinturier, L. PowerAPI: A Software Library to Monitor the Energy Consumed at the Process-Level, ERCIM News (2013).

https://www.semanticscholar.org/paper/PowerAPI%3A-A-Software-Library-to-Monitor-the-Energy-Bourdon-Noureddine/d57705a1bb4f4396f39d56ba0a5c2ab98b153c53 (accessed October 27, **2024**).

71. Discontinued: Intel® Power Gadget, Intel (n.d.). https://www.intel.com/content/www/us/en/developer/archive/tools/power-gadget.html (accessed November 1, **2024**).
72. Noureddine, A. PowerJoular and JoularJX: Multi-Platform Software Power Monitoring Tools, in: 2022 18th International Conference on Intelligent Environments (IE), **2022**: pp. 1–4. https://doi.org/10.1109/IE54923.2022.9826760.
73. Ahmad, R.W. ; Hamid, S.H.A.; Gani, A.; Obaidat, M.S.; Shuja, J.; Rehman, F.; Khan, A.U.R. Performance Assessment of Dynamic Analysis Based Energy Estimation Tools, in: 2018 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), **2018**: pp. 1–12. https://doi.org/10.1109/SPECTS.2018.8574182.
74. Acar, H.; Alptekin, G.I.; Gelas, J.-P.; Ghodous, P. The Impact of Source Code in Software on Power Consumption, Lyon, **2016**.
75. Oliveira, W.; Torres, W.; Castor, F.; Ximenes, B.H. Native or Web? A Preliminary Study on the Energy Consumption of Android Development Models, in: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), **2016**: pp. 589–593. https://doi.org/10.1109/SANER.2016.93.
76. Şanlıalp, I.; Ozturk, M.; Yiğit, T. Energy Efficiency Analysis of Code Refactoring Techniques for Green and Sustainable Software in Portable Devices, Electronics 11 (**2022**) 442. https://doi.org/10.3390/electronics11030442.
77. Chowdhury, S.A.; Hindle, A. GreenOracle: estimating software energy consumption with energy measurement corpora, in: Proceedings of the 13th International Conference on Mining Software Repositories, ACM, Austin Texas, **2016**: pp. 49–60. https://doi.org/10.1145/2901739.2901763.
78. Pang, C.; Hindle, A.; Adams, B.; Hassan, A.E. What Do Programmers Know about Software Energy Consumption?, IEEE Software 33 (**2016**) 83–89. https://doi.org/10.1109/MS.2015.83.
79. Bashroush, R.; Woods, E.; Noureddine, A. Data Center Energy Demand: What Got Us Here Won't Get Us There, IEEE Software 33 (**2016**) 18–21. https://doi.org/10.1109/MS.2016.53.
80. Pinto, G.; Castor, F. Energy efficiency: a new concern for application software developers, Commun. ACM 60 (**2017**) 68–75. https://doi.org/10.1145/3154384.
81. Wysocki, W. Why Don't Software Companies Care About Software Energy Efficiency? A Survey of Software Industry Developers., Procedia Computer Science 226 (**2024**).
82. Zhang, C.; Hindle, A.; German, D.M. The Impact of User Choice on Energy Consumption, IEEE Software 31 (**2014**) 69–75. https://doi.org/10.1109/MS.2014.27.
83. The EU Code of Conduct for Data Centres – towards more innovative, sustainable and secure data centre facilities - European Commission, (**2024**). https://joint-research-centre.ec.europa.eu/jrc-news-and-updates/eu-code-conduct-data-centres-towards-more-innovative-sustainable-and-secure-data-centre-facilities-2023-09-05_en (accessed October 31, **2024**).
84. Data Center Maturity Model | The Green Grid, (n.d.). https://www.thegreengrid.org/en/resources/library-and-tools/438-Data-Center-Maturity-Model (accessed October 31, **2024**).
85. Marat, A. Energy storage – intelligent energy management on the example of Automatic System Engineering., Er (**2022**) 69–74. https://doi.org/10.7494/er.2022.8.69.