2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Society Track

Awakening Awareness on Energy Consumption in Software Engineering

Erik Jagroep, Jordy Broekman, Jan Martijn E.M. van der Werf, Sjaak Brinkkemper
Utrecht University,
Dept. of Information and Computing Sciences
Princetonplein 5,
3584 CC Utrecht, The Netherlands
Email: {e.a.jagroep, j.broekman, j.m.e.m.vanderwerf, s.brinkkemper}@uu.nl

Patricia Lago
Vrije Universiteit Amsterdam,
Computer Science Institute
De Boelelaan 1081a,
1081 HV Amsterdam, The Netherlands
Email: p.lago@vu.nl

Leen Blom, Rob van Vliet
Centric Netherlands B.V.
P.O. Box 338,
2800 AH Gouda, The Netherlands
Email: {leen.blom, rob.van.vliet}@centric.eu

***Abstract*—Software producing organizations have the ability to address the energy impact of their ICT solutions during the development process. However, while industry is convinced of the energy impact of hardware, the role of software has mostly been acknowledged by researchers in software engineering. Strengthened by the limited practical knowledge to reduce the energy consumption, organizations have less control over the energy impact of their products and lose the contribution of software towards energy related strategies. Consequently, industry risks not being able to meet customer requirements or even fulfill corporate sustainability goals.**

In this paper we perform an exploratory case study on how to create and maintain awareness on an energy consumption perspective for software among stakeholders involved with the development of software products. During the study, we followed the development process of two commercial software products and provided direct feedback to the stakeholders on the effects of their development efforts, specifically concerning energy consumption and performance, using an energy dashboard. Multiple awareness measurements allowed us to keep track of changes over time on specific aspects affecting software development. Our results show that, despite a mixed sentiment towards the dashboard, changed awareness has triggered discussion on the energy consumption of software.

***Keywords*-Energy consumption perspective; Awareness; Software energy consumption; Software engineering;**

I. Introduction

Software is acknowledged by academia to be a key driver for the energy consumption of Information and Communication Technology (ICT) solutions [1], [2]. Software and energy, i.e., the area of green software [2], can be related to the environmental dimension of sustainability, which is generally defined as 'the capacity to endure' [3]. One way to address energy consumption in software, i.e. Software Energy Consumption (SEC), is through its software architecture [4]. By applying an Energy Consumption Perspective (ECP) [5], the SEC can be addressed in the early stages of software engineering. In this way, sustainability can be considered as a Quality Attribute [6], and consequently, it can be included in trade-off analysis and architecture evaluation [7]. However, a necessary prerequisite is that the stakeholders involved in the development are aware of the energy consumed by their software and its causes [8].

Different studies have surfaced investigating means to measure SEC [9], [10] compare releases of software products on their energy consuming characteristics [8], [11], and provide insight to those involved in software development [12]. Another study [13] reports limited knowledge of energy efficiency and lack of knowledge of practices to reduce the SEC among software developers. Additionally, the study reports uncertainty about how software consumes energy, which seems to contrast the findings of [14], that practitioners are aware of energy consumption problems. From these studies, it becomes clear that a gap remains with respect to concrete coding guidelines and practices reaching their target audience [15]. In other words, we see that industry is not yet able to adopt solutions provided by research.

With the growing attention for corporate social responsibility, Software Producing Organizations (SPOs) [16], such as independent software vendors and open-source foundations, risk not being able to fulfill their corporate sustainability goals and meet (customer) sustainability requirements with their software [8]. Awareness of their software products' energy consumption potentially helps SPOs to mitigate this risk.

In this paper, we present the findings of a multiple-case study on creating and maintaining awareness of the energy consumption of software products among stakeholders during development, as it has the greatest impact [17]. We first introduce an energy dashboard for the ECP based on earlier research [18], which provides insight in the energy consumption between consecutive sprints. For two commercial software product, we then measure how the awareness changed over several sprints. To measure the development of the awareness we create a specialized awareness model for SEC, inspired by the work of [19] and that served as a basis for the surveys held with the stakeholders after each sprint.

The main contribution of this paper is twofold:

- **Awareness model for SEC:** The model we apply allows us to capture the awareness of stakeholders involved with product development. The scores we obtain allow for analysis on different constructs, which provide insight

978-1-5386-2673-3/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSE-SEIS.2017.10

into the areas that require extra attention in relation to SEC and ECP.

- **Development of awareness over sprints:** Although the stakeholders are better aware of the ECP of their software products, the results show and shortcomings in current state-of-the-art tactics and best-practices to improve software energy efficiency. Furthermore, to maintain awareness, SEC should be supported throughout the whole organization.

The paper is structured as follows: Section II presents our research questions followed by a discussion of related work (Section III) and the design of our empirical study (Section IV). In Section V we present the results of our study which are discussed in Section VI, followed by the threats to validity (Section VII). Concluding remarks and an outline for future work are provided in Section VIII.

II. Research Questions

To address the issue presented above, our study was structured around the following **main research question (RQ)**:

RQ: *How to create and maintain awareness of the energy consumption perspective in software product development?*

Following 'A Dictionary of Psychology'[1] awareness is part of being 'conscious' that is: "giving due weight to something". In our context, being aware means weighted decisions can be made with respect to SEC, without implying an improvement or deterioration of the SEC. For an SPO, systematically addressing the SEC in the software design requires that awareness is maintained among the stakeholders involved with the software.

As a prerequisite to answer our RQ we need to be able to determine awareness among stakeholders, which leads to our first research sub-question:

SQ1: *How can we measure awareness on the topic of SEC?*

For this sub-question (SQ1) we look into those aspects that determine awareness and operationalize these in the software engineering context.

Second, to actually create awareness, we require a means to stimulate the stakeholders to actively think about SEC that can be incorporated in the development process. Resulting in the second and third sub-questions:

SQ2: *What stimulus can be used to trigger SEC awareness?*

SQ3: *How can we incorporate SEC in the development process?*

The second sub-question (SQ2) is set to investigate what information is required by the stakeholders and in which form the information should be presented. After determining what stimulus is required, we answer the final sub-question (SQ3) by investigating how the stimulus can be included in the development process, with minimal impact on the process itself, and enable stakeholders to structurally consider the SEC of their software products.

[1]http://www.oxfordreference.com/view/10.1093/acref/9780199534067.001.0001/acref-9780199534067

III. Background

Green Software: Green software generally refers to energy efficient software. To create green software, the sustainability aspect should be addressed during the early development stages of a product and constantly monitored during the software product lifecycle [17]. For example, applying green practices [15] and selecting the right 'Collection types' [20] potentially reduce the energy consumption up to respectively 25% and 300%. If we position sustainability as a software quality property [3], [5] we can go back further in the lifecycle, i.e. the design phase, where the software architecture allows for precluding qualitative traits of the software [4]. Similar to technical debt [21], early awareness of green software could save a significant amount of costs compared to refactoring the software at a later stage.

Energy Profiling: A recurring theme with green software is monitoring the SEC to support engineers with understanding their code and its energy impact [22]. However, unlike example the mobile domain [23], monitoring is more difficult with software products [1], [8] and different approaches exist to estimate the SEC. Most prominent are power models that profile the software based on resource usage, e.g. [9], [10], but new approaches are surfacing using big data principles [24]. A modeling specialist, for example, could help in building predictive profiling models [25]. In practice, performance is often used as a proxy for energy efficiency; i.e. less resource usage equals to less energy consumption. However, energy consumption and performance are not always positively correlated [26]–[28] and should thus be considered separately.

SEC Awareness: A lack of knowledge on SEC [13] does not imply green software should be neglected altogether. Knowing the difference in energy consumption between releases, e.g. [8], could help practitioners determine whether the energy consumption is reasonable given the work being performed [22]. Being aware of the topic, which fits the 'service awareness' problem area [29], could affect the believes of software engineers that are bound to affect their practice [30]. Following the economic dimension of sustainability [3], software-oriented data analytics [25] provides actionable insights to achieve business goals using green software practices.

Creating awareness requires a stimulus that triggers stakeholders to actively make conscious decisions with respect to SEC. Examples like the 'Eco' programming model [31], Resource Utilization Score (RUS) [18] and a graphical energy monitoring interface [12] have shown positive effects in this regard. However, creating awareness does not automatically imply a reduced energy consumption. A conscious decision could be to favor a specific quality aspect above sustainability, e.g. color usage to improve the usability [14]. In this case a conscious design trade-off is made [5].

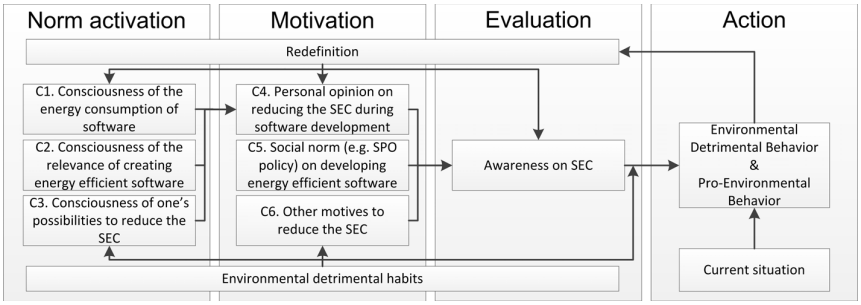


Figure 1. Model on transforming environmental behavior with the constructs translated to our study, after Matthies [19].

IV. Research Design

To answer our main research question we have conducted an embedded multiple-case study [32] where we measure SEC awareness and stimulus acceptance (i.e. multiple units of analysis) with two cases in a company developing commercial software products. In this section we describe the design of our study following the guidelines provided in [32]–[36].

A. Energy Consumption and Performance Measurements

To perform energy consumption measurements we applied a software-based approach using Microsoft Joulemeter (JM), similar to the approach applied in our earlier research [5], [8]. After calibration, JM allows us estimate the total energy consumed by a system at run time based on the computational resources used with a one second interval between measurements. To determine the SEC we subtract the idle energy consumption of a system from the energy consumed while running the software, both obtained using JM. The difference is in the energy consumed on the account of the software product, i.e. its SEC.

A prerequisite for valid measurement is to let the hosting server cool down to a stable state (i.e. a state with no active processes without direct instructions from the user [8]) after a reboot. The cooldown time has to be determined for each individual server used in the study.

In turn, the performance of the hardware components was measured using Windows Performance Monitor, a standard tool with the Windows operating system. Developers are in general more familiar with performance aspects, e.g. CPU utilization, and have experience with addressing the performance aspects of a system. To provide insight in the resource usage by the software we calculate a Resource Utilization Score, or RUS [18] – a score for the software based on the relevant performance aspects according to the stakeholders.

SEC and Performance Measurements Protocol: To ensure the validity of the SEC measurements, a simple protocol was followed to perform each run:

1) Restart the environment.
2) Close unnecessary applications.
3) Start performance measurements and setup JM.
4) Remain idle for the duration of the cooldown time.
5) Start JM measurements.
6) Start load test and wait for test to finish.
7) Collect and check data.

Starting the performance measurements upfront (step (3)), allowed us to check whether the system was indeed in a stable state during a run. If this was not the case (checked in step (7)), the run was excluded from further analysis.

B. On measuring awareness on SEC

In designing the study to answer SQ1, we found that 'awareness' cannot be measured directly due to the inability to quantify the concept [37]. Hence, as a means of indirect measurement, we used the model presented by Matthies [19] meant to transform environmental-detrimental habits into pro-environmental habits, and specialized it to capture changes in awareness on SEC. The resulting model (Fig. 1) consists of four stages (norm activation, motivation, evaluation and action) and six constructs (C1 through C6) that directly or indirectly affect the weighting of moral, social and other types of costs and benefits, potentially resulting in a behavioral change.

Following the definition of awareness provided in Section II, for stage Evaluation we relabeled the activity of 'weighting relevant aspects' into 'awareness on SEC' and defined a survey to measure it. We used the specialized constructs as basis for defining the survey questions[2]. In the Action stage of the model we determine whether behavior has indeed changed.

As illustrated in Table I, we formulated 14 statements based on the constructs. To this aim, we carried out brainstorming sessions with experts in the field of green software engineering (including the authors) combined with related works like [13], [38]. While the statements are related to personal experience, it has been recognized that such personal experience has the strongest influence on beliefs with respect to specific topics related to software engineering [30]. (See also Section VII for a discussion of the related threats to validity.) Each individual statement can be answered with an option ranging from strongly disagree (-2), disagree (-1), neutral (0), agree (1) to strongly agree (2) – where the number behind every option represents our internal coding scheme.

Next to measuring SEC awareness, the survey also includes statements for measuring the acceptance of the awareness stimulus (i.e. the dashboard, see Section IV-C). To this aim,

[2]The complete survey is available online at http://tinyurl.com/gvpf3hg.

Table I
THE STATEMENTS PER CONSTRUCT USED FOR THE AWARENESS MEASUREMENTS, WITH ITS PARTITIONING INTO SURVEYS A AND B.

C1 (A)	1	I want to determine the energy consumption of our software development environment (e.g. by calculating the energy consumed by (test)servers, laptops and other resources).
	2	Before this project, I wondered multiple times about the energy consumption of our software development environment.
C2 (B)	3	I expect that software has a large influence on the energy usage.
	4	I would like to know the energy consumption of our software product.
C3 (B)	5	If I had more time to work on the code, I would be able to reduce the energy consumption.
	6	The applied techniques (programming language, design patterns, etc.) to realize the software product, allow for the reduction of energy consumption.
	7	It is possible to make a trade-off between our current non-functional requirements (e.g. performance) and the energy consumption of our software.
C4 (A)	8	Addressing the energy consumption of our software should gain more attention.
	9	I would like to reduce the energy consumption, if I am allowed to spend time on it.
C5 (A)	10	The energy consumption of our software product is discussed during (in)formal meetings.
	11	If other teams reduce the energy consumption of their software, I would attempt it too.
	12	Reducing the energy consumption would be a benefit to the organization and the customer.
C6 (A)	13	Code optimizations to improve non-functional requirements should be acknowledged and included in our backlog.
	14	The benefits of rewriting the code to reduce the energy exceed the costs.

we used as basis constructs inspired by the Unified Theory of Acceptance and Use of Technology (UTAUT) [39], [40], and included the relevant associated statements. Both constructs and associated statements are shown in Fig. 6.

The first version of the survey was reviewed by ten practitioners in our network (software engineers and IT managers) with varying years of experience. The main feedback was related to the formulation of the statements. However, several warnings were also issued with respect to the length of the survey. As we intended to present the survey to stakeholders multiple times (see Section IV-D), we have split the survey in three separate sections:

- Survey A, evaluating the awareness with a generic and broad scope (C1 and C4-C6).
- Survey B, evaluating the awareness related to a specific software release (C2 and C3).
- Survey C, focused on the acceptance of the awareness stimulus (in our case, the dashboard).

Each survey section is presented to the participants only when relevant, as exemplified in Fig. 3. This allowed us throughout the study to distribute the effort required to the participants while collecting all necessary data.

C. The Stimulus to Trigger SEC Awareness

To answer SQ2, we used as input the work done in [18] and [12], and combined them resulting in an energy dashboard, as shown in Fig. 2. This includes a radar chart and an overview of the exact measurements. Data for the dashboard is obtained by following the provided measurement protocol (Section IV-A).

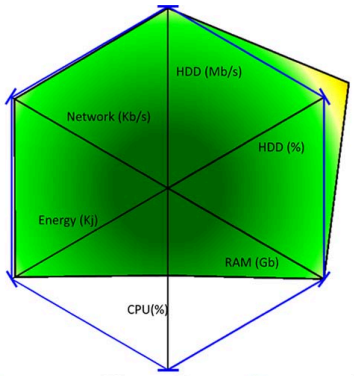
In particular, the *radar chart* graphically shows the RUS [18] and is meant to enhance the communication of the measurements to stakeholders and highlight key findings. As adopted in [12], visual information coveys familiar indicators and gauges, e.g. percentages and bar charts, that 'non-energy experts' can easily grasp. Regarding the *individual measurements*, the energy dashboard (lower part of Fig. 2) includes the *delta* between two releases, which indicates the change with respect to the usage of a specific resource [18]. Calculating deltas requires labeling one release as benchmark and positioning the measurements of a different release in light of this benchmark. In the example of Fig. 2, the results show a decrease in energy consumption with 2.15% of the new release (black line) compared to the previous release (blue line), whereas the color of the surface (green, to yellow to red) represent the intensity of the decrease (green) or increase (red) in resource usage. With multiple consecutive releases, each new release is set as the benchmark for the next, thereby summarizing the effects of the latter release.

D. On incorporating the Stimulus in Development Process

To answer SQ3, we followed the advice of Devanbu et al. [30] to take practitioners' beliefs into account in designing an experiment. A short investigation with multiple development teams and the experience of the authors learned that Scrum was the common development method in the company, and that software-related dashboards were frequently consulted at the end of each sprint. At this point in time the team typically reflects on the past sprint and decides on corrections, e.g. (re-)prioritize requirements, if required. Accordingly, it was natural to present our dashboard shortly after each sprint.

Fig. 3 illustrates the holistic organization of our multiple-case study. The survey and energy dashboard have been incorporated in the Scrum development process used by the involved company. **At the start**, survey A and B are presented to determine the initial awareness of the stakeholders, followed by the **preparation** phase where the first two releases of a software product (r.1 and r.2) are tested. The preparation phase ends with the sprint review for release 2 where the first dashboard (dashboard r.2 - r.1) is presented. The order of releases with the dashboard indicates that, e.g., release 2 is compared to release 1, and as such release 1 served as the benchmark for calculating the delta's.

After the preparation phase, we repeat the following procedure for each consecutive sprint. At the end of a sprint, while the new release is being tested by the team, the load test can be performed to collect data for the energy dashboard. In the accompanying sprint review the stakeholders fill in



Version	CPU (%)	RAM (Gb)	HDD (%)	HDD (Mb/s)	Network (Kb/s)	Energy (Kj)	RUS
5.3.12	14.10	1.80	13.64	2.24	89.36	27.03	6424
5.3.13	6.77	1.80	15.81	2.22	87.76	26.45	5543
Delta	-51.99%	+0.00%	+15.91%	-0.89%	-1.79%	-2.15%	-13.71%

Figure 2. The energy dashboard: example as presented to the stakeholders of case RS.

survey B and C, looking back at the past sprint and dashboard presented at the previous sprint review. Afterwards the new dashboard is presented. In the last iteration, survey A, B and C are presented, followed by the final dashboard. The **case evaluation** with the team closes a case and helps to determine whether behavior has changed (i.e. the Action stage of the model in Fig. 1).

E. Case Selection

The cases included in our study were acquired by contacting product managers of multiple software products within the case company; a large international SPO. While we did not have specific criteria for the software products themselves, we did formulate the following inclusion criteria related to the processes, technology, and team:

- An agile development method is implemented including reviews after each development iteration.
- At least six releases are available, four of which will be developed during the case study.
- The software product can be deployed in a production-like environment.
- Automated load tests are available.
- Commitment to participate for the duration of at least five sprints including filling in multiple surveys.

With respect to the load test, we are striving for usage scenarios as these are used most often when practitioners evaluate energy usage [22]. In the end we identified two cases that met all inclusion criteria. The details of these cases are provided below.

Case 1: Document Generator (DG) is a commercial software product used by over 300 (mostly governmental) organizations in the Netherlands, counting more than 900 end-users, and generating more than 30 million documents on an annual basis. Although DG has been used in earlier research [5], [8], meanwhile the product has moved to a new development team. This assures us of not having biased results with respect to awareness. We identified four developers and one tester as the stakeholders involved with developing DG and commitment was given for releases 8.0.6, 8.0.7, 8.0.7.1, 8.0.8, 8.0.9 and 9.0.0 being developed during the study. The duration of a sprint was three weeks, which meant the case study would last for fifteen weeks.

DG was deployed in a production-like environment encompassing an application and database server, both with a cooldown time of 20 minutes. The load test was designed to simulate a user generating 258 complex documents. Within the limited time for testing we were aiming for at least 40 measurements per DG release. Discussing the measurements with our DG contact learned that, apart from energy consumption, the CPU utilization, memory usage, hard disk usage, network usage and execution time were of interest.

Case 2: Retail System (RS) is a commercial software product for retail stores, e.g. supermarkets, to process the customer transactions of their points of sales, e.g. cash registers. With a customer base of 110 customers in 30 countries, counting more than 20.000 stores and 75.000 points of sales, RS processes more than 20 billion transactions on an annual basis. For RS, the 23 stakeholders participating in our study were located in Belgium and Romania: 16 developers, 1 database developer, 2 technical analysts, 2 testers, 1 software architect and 1 product specialist. Commitment was given for releases 3.11, 3.12, 3.13, 3.14, 3.15 and 3.15.1. Each sprint takes three weeks, resulting in a total case duration of fifteen weeks.

The RS software was deployed on a single server, with a cooldown time of 15 minutes, which corresponds to the most simple production-like setting. Despite its simplicity, our load test was designed to simulate the transactions for up to 100 points of sales, i.e. multiple supermarkets, in a fixed time-span of three hours. Considering the limited time window, we aimed to perform at least 10 runs for each RS release. With respect to the measurements, our RS contact pointed out the CPU utilization, memory usage, hard disk usage and network usage should be measured.

F. Data Analysis Procedure

The number of parcticipants per case (n=5 for DG, and n=22 for RS), led us to follow a qualitative approach to analyze our data [41] using awareness (survey A and B) and acceptance (survey C) scores calculated using the survey results. We calculate scores by adding up individual scores into a statement score, adding up the statement scores to score a construct, and finally adding up the construct scores resulting in a score for awareness and acceptance. Following the coding scheme of our data, i.e. from -2 to +2, a negative score indicates disagreement with the statements and vice versa. To accurately show changes over time, we only include the results of participants that filled in survey A at the start and end of the study and missed at most one combination of survey B and C.

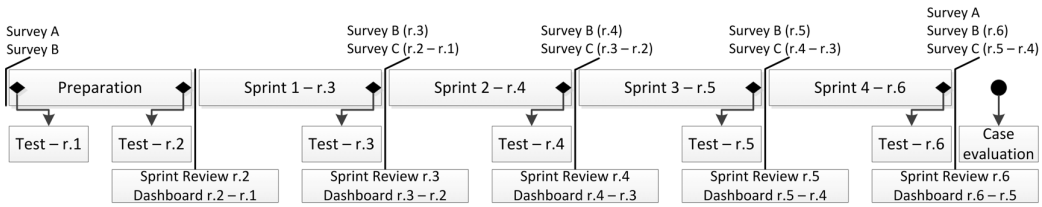


Figure 3. The case study organization including sprints, test periods, sprint reviews, energy dashboard presentations and surveys.

V. Results

In this section we report on the execution and the results of our multi-case study, for both cases. Detailed figures including scores per statement, like Fig. 6, are provided online[3].

A. Study Execution

With DG we deviated from the case description by excluding release 9.0.0 from the study. Even with support of the team, successful deployment was not possible and, consequently, release 8.0.9 was the final release included in the study. Due to time constraints and the planning of the DG team we could not compensate for this exclusion, resulting in a data gap for sprint review r.5. With the included releases we managed to perform the required 40 valid runs and collect all survey data with the exception of survey B (r.3) and survey C (r.2 - r.1) of two team members due to holidays. Despite the missing data, following our analysis procedure, the input of all five team members was included in the score calculations.

With RS, given the geographical distribution, an online survey tool was used to conduct the surveys, with the following survey response rates: 96% (SR-r.1), 65% (SR-r.3), 61% (SR-r.4), 57% (SR-r.5) and 65% (SR-r.6). Following the data analysis procedure, the RS scores were calculated based on the survey results of twelve team members. Additionally, we managed to perform nine runs per release, instead of the required ten, due to sharing of the test resources and issues with Performance Monitor. The available resources for load testing were also used for production testing which simply had a higher priority. Investigation into Performance Monitor pointed out data is automatically deleted when a specific threshold is reached for the available hard disk space. An issue we solved with a simple reconfiguration. Despite missing one run we still obtained sufficient data per release to produce valid energy dashboards.

The evaluation for both cases took place approximately two weeks after 'sprint 4' with representatives of the team.

B. Survey A and B

The final score on awareness for the DG team decreased from +23 to +3 at the end of the case study. On construct level the results (Fig. 4) show an increase with respect to consciousness of the energy consumption of software (C1) and the social norm on developing energy efficient software (C5). On the other hand the stakeholders indicate that the relevance of creating energy efficient software (C2) has decreased together with the consciousness of one's possibilities to reduce the SEC (C3), the personal opinion on reducing the SEC (C4) and the other motives to reduce the SEC (C6). The scores on C2 and C3 (collected with survey B), resemble the patterns of the Gartner Hype Cycle with a change from strongly positive to strongly negative and back to a more neutral score, i.e. zero.

The awareness scores for RS changed from +4 to -16 during the case study, and on construct level (Fig. 5) resemble the trends found with DG. C2 and C3 again show the Gartner Hype Cycle pattern, C4 and C6 decrease over time with C4 even becoming negative and C5 changed from negative to a positive score. The only discrepancy is with C1 where RS stakeholders became more negative.

C. Survey C

With respect to the acceptance of the energy dashboard, in general the statement scores of the DG stakeholders (Fig. 6) are increasingly negative over sprint reviews – resulting in a change of the total score from 14 to -24. The only exception is with the effort expectancy (EE) construct, where the final score of +5 (obtained by adding up the EE scores for SR r.6) implies that the dashboard is considered user friendly. The attitude towards technology (AT) follows in a second place with a score moving from +6 to 0. Scores concerning the performance expectancy (PE), social influence (SI) and behavioral intention (BI) become increasingly negative, namely -3 to -5, -2 to -13 and +8 to -11 respectively.

For RS (Fig. 7), the total dashboard acceptance score shifts from -79 to -118 and we again only find a positive final score (+6) with the EE construct. The other construct scores imply a consistent negative acceptance of the energy dashboard resulting in final scores of -26 (AT), -35 (BI), -18 (PE) and -45 (SI).

VI. Discussion

In this section we discuss the results in light of our research sub-questions. To this aim, we follow a method similar to sentiment analysis [42].

A. SQ1: Measuring awareness on SEC

With respect to C1, the sentiment switched from negative (-2) to positive (+5) with DG (Fig. 4), indicating an increased willingness to determine the energy consumption in relation to software, and became more negative (-13 to -18) with RS (Fig. 5). The case evaluations learned that both teams were

[3]Scores per statement available at http://tinyurl.com/gvpf3hg.

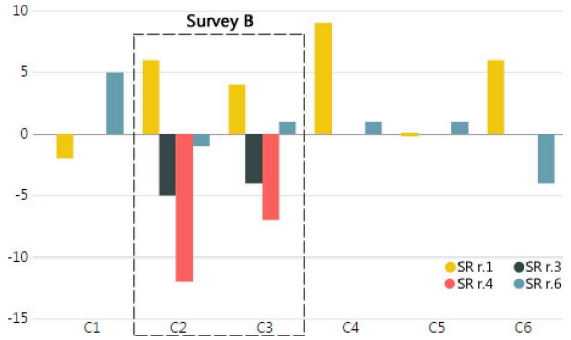


Figure 4. DG case: The awareness scores per construct over sprint reviews.

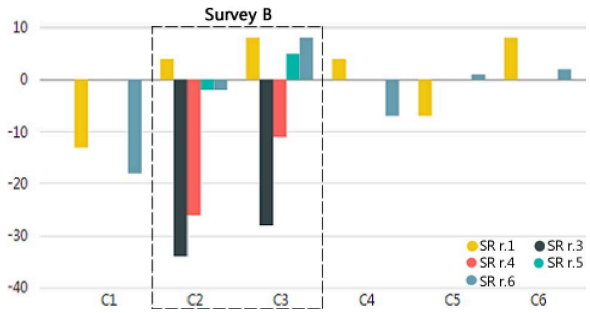


Figure 5. RS case: The awareness scores per construct over sprint reviews.

more aware of the SEC and would keep the topic in mind during development, hence implying a change in behavior. However, the focus remains on software functionality in line with product strategy and customer demand.

Software architecture: With C2 and C3, stakeholders from both cases increasingly want to know the SEC of their software product, which seems in contrast with the RS scores on C1. Investigating the individual statements for RS, we found a potential explanation in the negative score concerning the influence of software on energy consumption (i.e. -15 to -6). In contrast with DG, where the case ends with a neutral score on this statement. Rephrased, while there is an interest in the SEC, with RS there is doubt on the impact of the software on energy consumption. One RS developer stated from previous experiences with mobile software that the effectiveness of any effort is clouded by the different layers of the software stack (e.g. operating system and middleware). While the move towards a more neutral sentiment indicates a greater acknowledgement of the role of software, hardware and other aspects are still believed to have a greater impact on the energy consumption.

When asked, RS stakeholders appointed software architects as the main actors in relation to SEC along with hardware manufacturers, and a declining role for software developers. In contrast, DG stakeholders consistently appoint software developers and architects as the main actors. These findings provide a plausible reason for the sentiment on C3 and remained negative sentiment on specific statements for this construct. For example, given the large group of developers involved, the RS stakeholders maintain a negative sentiment towards the statement on spending more time to reduce the SEC. However, driving the positive C3 score, RS stakeholders acknowledge the importance of the applied techniques (+6) and trade-offs with non-functional requirements (+4).

The results for C6 also confirm the focus on software architecture. For example, stakeholders reconsidered the statement on including code optimizations to reduce the SEC in the backlog (+6 to -3 with DG, and +11 to +6 for RS). A possible cause is that the stakeholders increasingly positioned SEC in relation to non-functional requirements of the software [4], which are typically not included in the backlog. This finding could explain the stakeholders' negative sentiment towards the statement that the benefits of rewriting the code to reduce the SEC exceed the costs.

Learning curve: The statements of C3, on the extent to which the applied techniques and making trade-offs with non-functional requirements can contribute to reducing the SEC, suggest a learning curve took place. Starting with positive scores, stakeholders from both cases appear to underestimate the complexity involved with reducing the SEC. After the first dashboard presentation, the scores became negative with stakeholders realizing the complexity involved. The movement towards a neutral or even positive score follows after stakeholders better comprehend the SEC. Interesting to note is that DG stakeholders indicate the SEC is not discussed during (in)formal meetings (C5), whereas a lively discussion on SEC took place during multiple sprint reviews. The RS scores move from -17 to -8 on the latter statement, and the evaluation learned that SEC is discussed by specific groups, e.g. lead developers and architects, within the team.

SEC on the team agenda: Given the developments with C1, C2 and C3, a decline with C4 was expected. With DG a slight positive sentiment remains as the stakeholders are willing to address the SEC when allowed to spend time on this aspect. However, in line the discussion so far, RS becomes negative (+4 to -7) on this statement.

Stakeholders from both cases indicated extra attention towards SEC is only required when reduced energy consumption becomes a requirement from the customer or a strategic driver within the organization. Here we also find a difference between cases; the DG team could obtain a strategic advantage by being delivering a more 'sustainable' product, whereas RS did not identify this advantage within their markets. A finding that fits the survey results and potentially shows a difference between the markets involved in the study.

Willingness: The statement on seeing other teams address the SEC (C5) moves from a negative (-2) to a positive (+2) sentiment with DG and increases from +1 to +3 with RS. Additionally, both cases acknowledged that reducing the SEC benefits the organization and the customer, however customer demand and market trends lead in determining the future of the product. The results indicate a willingness to address the SEC when sustainability goals are formulated and clear benefits can be identified. However, even without strategic goals an SPO could benefit from promoting energy efficient

Figure 6. DG case: The scores on the individual acceptance statements (survey C), grouped per construct.

Figure 7. RS case: The acceptance scores per construct over sprint reviews.

software development. An overall reduction of the total cost of ownership for a product allows an SPO to, e.g., maintain a more competitive pricing model.

B. SQ2: Stimulus to trigger SEC awareness

Solely based on the acceptance results (cf. Fig. 6 and Fig. 7), our conclusion would be that the dashboard does not appear the right means to trigger awareness with respect to SEC: while the dashboard is easy to use, shown by the effort expectancy (EE), it does not stimulate to perform target behavior (AT) or to positively influence technology usage (BI).

Confronting: Interpreting the scores in light of the awareness scores, we can only partially explain the negative sentiment found in the scores. If stakeholders do not know how to reduce the SEC or become aware of the limitations imposed by, e.g., the technology used, presenting the dashboard could frustrate and result in a negative sentiment towards the dashboard. To this end, the dashboard does trigger awareness, be it in a confronting manner. This suggests that awakening awareness would be effective when combined with software engineering knowledge on how to decrease the SEC.

Despite potentially being confronting, 'Using the dashboard is a good idea' (AT) is one of few positive statements for RS. Also DG stakeholders indicate they like working with the dashboard and that the dashboard makes work more interesting. However, DG stakeholders did not find using the dashboard a good idea. While we cannot explain the unexpected decline with the latter, the results potentially indicate the dashboard to contain useful elements in this context.

Presentation frequency: Apart from being confronting, the frequency of presenting the dashboard, i.e. after every sprint, could also have contributed to the negative sentiment. For example, while the statement concerning the usefulness for the job starts positive in both cases, consecutive measurements indicate a negative sentiment. During the evaluation, stakeholders indicated that the SEC should not be inspected after every sprint, unless there is a concrete motive to do so. Instead, similar to other quality attributes, a periodical evaluation of the SEC is considered to be sufficient.

Target audience: with DG a contrasting finding is found with the PE statements (Fig. 6): while the dashboard is considered increasingly less useful for the job, stakeholders disagree less with using the dashboard as a means to increase programming skills. The diversity of roles included in the study, i.e. also non-developers, potentially explains this contrast, which would imply the dashboard in its current form is role-specific and should be tailored for its target audience. Combined with the SI results, indicating that stakeholders find little support with others in using the energy dashboard, an SPO should carefully consider what information to present on a dashboard and to whom.

C. SQ3: incorporating SEC in the development process

The survey results indicate that the stakeholders are able to make weighted decisions with respect to SEC, i.e. are aware of the topic. However, for an SPO the ability to fulfill corporate social responsibility goals with software products requires that awareness is maintained. Based on the results, we can provide the following related recommendations:

Formulate sustainability goals: Stakeholders indicated that a reduced energy consumption benefits the organization and the customer. When positioned as a strategic goal for the organization, stakeholders can justify efforts to address the SEC of their products. Additionally, by embedding SEC in the organization, an SPO can also experience benefits from the social norm that motivate stakeholders (C5). An effect that can be potentially amplified by means of, e.g., gamification.

Set up knowledge bank: The overall impression, confirmed by the results, is that there is limited knowledge on how to

actually address the SEC. Additionally, the results indicate stakeholder increasingly becoming aware of the possibilities in relation to SEC, which highlights the importance of having an ECP [5] to guide decision-making. A knowledge bank on this topic, containing e.g., tactics, patterns and best practices, provides concrete guidelines, and potentially makes green software practices more cost effective. Relating the knowledge bank to the views in the ECP helps to make trade-offs on the different aspects related to software design and strengthens the relation with sustainability goals.

Quantify SEC: Despite the limited acceptance, the energy dashboard proved useful to show the effects of development efforts. Quantification makes SEC more concrete, hence enabling informed decision making. Additionally, the energy dashboard can highlight specific achievements and potentially encourage development efforts to further improve the software quality. For example, the reduced CPU utilization by solving a long term bug in RS (see Fig. 2) was put in the spotlight using the energy dashboard. When an energy dashboard, or similar stimulus, is to be implemented, an SPO needs to keep the target audience in mind as this could greatly affect acceptance.

VII. Threats to Validity

This section presents the threats to validity as required by [33], [35], [36]. For our study, the threats to validity can be related to two separate aspects; performing SEC measurements and the case study itself. With respect to the former, we applied the methodology as described in [8] and as such were confronted with the same threats to validity. Specifically the reliability of JM, measurement interval, operating system effects, energy consumption overhead and measurement tooling were of concern, and countered in the same manner [8]. In this section, we focus on the latter aspect, the case study itself.

For the **internal validity**, uncontrolled factors that might affect our results, we acknowledge that stakeholders could have modified their behavior in response to being observed, i.e. the *Hawthorne effect*. Additionally, following the *principal-agent problem*, the motivation of the stakeholders to address SEC could be affected by their (in)ability to choose the technology being used and the fact that they do not pay the energy bill. To minimize the effects we respectively minimized the intrusiveness of our protocol and focused on the personal experience with our survey statements.

External validity addresses the extent to which the results can be generalized. While our *inclusion criteria* could exclude SPOs from participating in our study, we argue that RS and DG are representative for cases found in the software industry. However, we do acknowledge the limited ability to generalize findings beyond the two specific cases, and the limited *number of participants* should be considered to, at best, provide insight in SEC-related awareness in software engineering. Additionally, further investigation is required into *cultural differences*. Although the cases were separate enough within the case company, thereby not affected by company-wide cultural aspects, we were not able to investigate differences between countries which would require more case studies.

The **construct validity**, the degree to which the measures capture the concepts of interest, is focused around the survey. First, the *definition of awareness* allowed us to create a survey and perform the case study. However, as it is a field on its own, we do not aim to provide a general definition of the term. The resulting *survey statements* are systematically developed and valid proxies for awareness. Other statements might be included depending on the context, such as the software and the organization under study. To minimize this threat we carefully designed our study around the model in Fig. 1, and established a chain of evidence that allowed us to relate the individual statements to the six awareness constructs.

Finally, **reliability** considers the extent to which research is dependent on the specific researchers. By describing, in detail, the *protocol* that was followed as well as the forthcoming analysis, we provide openness in the research that is performed. Deviations from the protocol were described as such. With respect to the sentiment analysis being researcher dependent, we support our claims with the data obtained from the study.

VIII. Conclusions

In this paper we present the results of a embedded multi-case study performed with two cases regarding two commercial software products. To provide an answer to our main research question "how to create and maintain awareness of the energy consumption perspective in software product development?", we first investigated three sub-research questions.

To measure awareness (SQ1) we constructed a survey based on six constructs that directly and indirectly affect awareness. Although the survey is specific for our purposes and by no means a generic solution, the survey data allowed us to express awareness using a score. As a stimulus to trigger awareness (SQ2), an energy dashboard was created including those measurements the stakeholders consider relevant in relation to the software product in their daily practice. The survey and dashboard were combined in the overall multi-case study organization, hence incorporating SEC awareness creation and the related impact in the software development process (SQ3).

With respect to creating awareness, we found the stakeholders of both cases to actively discuss the topic during the case study and able to make weighted decisions with respect to SEC. In other words, appropriate stimuli help *awaken SEC awareness*. To *maintain SEC awareness*, our results show that organizational policy is required to support creating green software products strengthened with a knowledge bank to stimulate informed decision making on software design.

In future work, we plan to automate the testing activity in case study organization to further lower the threshold to perform SEC measurements. Also we plan to investigate the SEC with individual development efforts, e.g. commits, to distill guidelines for green software knowledge banks.

Acknowledgments

We would like to thank the DG and RS teams for their participation in our research, and Fabiano Dalpiaz, Garm Lucassen, Başak Aydemir and Matthieu Brinkhuis for their valuable discussions and feedback to improve the paper.

REFERENCES

[1] A. Hindle, "Green mining: a methodology of relating software change and configuration to power consumption," *Empirical Software Engineering*, pp. 1–36, 2013.

[2] P. Lago, R. Kazman, N. Meyer, M. Morisio, H. A. Müller, F. Paulisch, G. Scanniello, B. Penzenstadler, and O. Zimmermann, "Exploring initial challenges for green software engineering: summary of the first GREENS workshop, at ICSE 2012," *ACM SIGSOFT Software Engineering Notes*, vol. 38, no. 1, pp. 31–33, 2013.

[3] P. Lago, S. A. Koçak, I. Crnkovic, and B. Penzenstadler, "Framing sustainability as a property of software quality," *Commun. ACM*, vol. 58, no. 10, pp. 70–78, sep 2015.

[4] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, ser. SEI Series in Software Engineering. Pearson Education, 2012.

[5] E. Jagroep, J. M. van der Werf, S. Brinkkemper, L. Blom, and R. van Vliet, "Extending software architecture views with an energy consumption perspective," *Computing*, pp. 1–21, 2016.

[6] ISO, "Systems and software engineering – systems and software quality requirements and evaluation (SQuaRE) – system and software quality models," International Organization for Standardization, Geneva, Switzerland, ISO 2510:2011, 2011.

[7] R. Kazman, M. Klein, M. Barbacci, and T. Longstaff, "The architecture tradeoff analysis method," in *4th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS) 1998*. IEEE, 1998, pp. 68–78.

[8] E. A. Jagroep, J. M. E. M. van der Werf, S. Brinkkemper, G. Procaccianti, P. Lago, L. Blom, and R. van Vliet, "Software energy profiling: Comparing releases of a software product," in *Proceedings of the 38th International Conference on Software Engineering Companion*, ser. ICSE '16. New York, NY, USA: ACM, 2016, pp. 523–532.

[9] E. Jagroep, J. M. E. M. van der Werf, S. Jansen, M. Ferreira, and J. Visser, "Profiling energy profilers," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. ACM, 2015, pp. 2198–2203.

[10] A. Noureddine, R. Rouvoy, and L. Seinturier, "Monitoring energy hotspots in software," *Automated Software Engineering*, pp. 1–42, 2015.

[11] A. Hindle, A. Wilson, K. Rasmussen, E. J. Barlow, J. C. Campbell, and S. Romansky, "Greenminer: A hardware based mining software repositories software energy consumption framework," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: ACM, 2014, pp. 12–21.

[12] A. Noureddine, S. Islam, and R. Bashroush, "Jolinar: Analysing the energy footprint of software applications (demo)," in *The International Symposium on Software Testing and Analysis*, ser. Proceedings of the 25th International Symposium on Software Testing and Analysis, Saarbrücken, Germany, Jul 2016, pp. 445–448.

[13] C. Pang, A. Hindle, B. Adams, and A. E. Hassan, "What do programmers know about software energy consumption?" *IEEE Software*, vol. 33, no. 3, pp. 83–89, May 2016.

[14] G. Pinto, F. Castor, and Y. D. Liu, "Mining questions about software energy consumption," in *Working Conference on Mining Software Repositories*, ser. MSR. New York, NY, USA: ACM, 2014, pp. 22–31.

[15] G. Procaccianti, H. Fernàndez, and P. Lago, "Empirical evaluation of two best practices for energy-efficient software development," *Journal of Systems and Software*, vol. 117, pp. 185–198, 2016.

[16] L. Xu and S. Brinkkemper, "Concepts of product software," *European Journal of Information Systems*, vol. 16, no. 5, pp. 531–541, 2007.

[17] S. Naumann, M. Dick, E. Kern, and T. Johann, "The greensoft model: A reference model for green and sustainable software and its engineering," *Sustainable Computing: Informatics and Systems*, vol. 1, no. 4, pp. 294–304, 2011.

[18] E. Jagroep, J. M. E. M. van der Werf, J. Broekman, S. Brinkkemper, L. Blom, and R. van Vliet, "A resource utilization score for software energy consumption," in *Proceedings of the 4th International Conference ICT for Sustainability*, ser. Advances in Computer Science Research. Amsterdam, The Netherlands: Atlantis Press, 2016.

[19] E. Matthies, "How can psychologists better put across their knowledge to practitioners? suggesting a new, integrative influence model of pro-environmental everyday behaviour," *Umweltpsychologie*, vol. 9, no. 1, pp. 62–81, 2005.

[20] S. Hasan, Z. King, M. Hafiz, M. Sayagh, B. Adams, and A. Hindle, "Energy profiles of java collections classes," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: ACM, 2016, pp. 225–236.

[21] L. Xiao, Y. Cai, R. Kazman, R. Mo, and Q. Feng, "Identifying and quantifying architectural debt," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: ACM, 2016, pp. 488–498.

[22] I. Manotas, C. Bird, R. Zhang, D. Shepherd, C. Jaspan, C. Sadowski, L. Pollock, and J. Clause, "An empirical study of practitioners' perspectives on green software engineering," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: ACM, 2016, pp. 237–248.

[23] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof," in *Proceedings of the 7th ACM european conf. on Computer Systems*, ser. EuroSys '12. New York, NY, USA: ACM, 2012, pp. 29–42.

[24] S. A. Chowdhury and A. Hindle, "Greenoracle: Estimating software energy consumption with energy measurement corpora," in *Proceedings of the 13th International Conference on Mining Software Repositories*, ser. MSR '16. New York, NY, USA: ACM, 2016, pp. 49–60.

[25] M. Kim, T. Zimmermann, R. DeLine, and A. Begel, "The emerging role of data scientists on software development teams," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: ACM, 2016, pp. 96–107.

[26] A. E. Trefethen and J. Thiyagalingam, "Energy-aware software: Challenges, opportunities and strategies," *Journal of Computational Science*, vol. 4, no. 6, pp. 444 – 449, 2013, scalable Algorithms for Large-Scale Systems Workshop (ScalA2011), Supercomputing 2011.

[27] K. K. Rangan, G.-Y. Wei, and D. Brooks, "Thread motion: Fine-grained power management for multi-core systems," *SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 302–313, 2009.

[28] T. Cao, S. M. Blackburn, T. Gao, and K. S. McKinley, "The yin and yang of power and performance for asymmetric hardware and managed software," in *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ser. ISCA '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 225–236.

[29] P. Lago and T. Jansen, "Creating environmental awareness in service oriented software engineering," in *Service-Oriented Computing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6568, pp. 181–186.

[30] P. Devanbu, T. Zimmermann, and C. Bird, "Belief & evidence in empirical software engineering," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: ACM, 2016, pp. 108–119.

[31] H. S. Zhu, C. Lin, and Y. D. Liu, "A programming model for sustainable software," in *International Conference on Software Engineering - Volume 1*, ser. ICSE. IEEE Press, 2015, pp. 767–777.

[32] R. Yin, *Case Study Research: Design and Methods*, ser. Applied Social Research Methods. SAGE Publications, 2009.

[33] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslé n, *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

[34] B. Kitchenham, H. Al-Khilidar, M. Babar, M. Berry, K. Cox, J. Keung, F. Kurniawati, M. Staples, H. Zhang, and L. Zhu, "Evaluating guidelines for reporting empirical software engineering studies," *Empirical Software Engineering*, vol. 13, no. 1, pp. 97–121, 2008.

[35] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[36] N. Juristo and A. M. Moreno, *Basics of Software Engineering Experimentation*, 1st ed. Springer, 2010.

[37] T. Nagel, "What is it like to be a bat?" *The Philosophical Review*, vol. 83, no. 4, pp. 435–450, 1974.

[38] R. E. Dunlap, "The new environmental paradigm scale: From marginality to worldwide use," *The Journal of Environmental Education*, vol. 40, no. 1, pp. 3–18, 2008.

[39] V. Viswanath, M. G. Morris, G. B. Davis, and F. D. Davis, "User acceptance of information technology: Toward a unified view," *MIS Quarterly*, vol. 27, no. 3, pp. 425–478, 2003.

[40] M. D. Williams, N. P. Rana, Y. K. Dwivedi, and B. Lal, "Is utaut really used or just cited for the sake of it? a systematic review of citations of utaut's originating article." in *ECIS 2011 Proceedings*, 2011.

[41] M. B. Miles and A. M. Huberman, *Qualitative data analysis: An expanded sourcebook*. Sage, 1994.

[42] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Found. Trends Inf. Retr.*, vol. 2, no. 1-2, pp. 1–135, Jan. 2008.