Green Software: The Awareness Awakens

MBI Thesis

J.M. Broekman

Utrecht University

January 2017

"Measure what is measurable, and make measurable what is not so"

Galileo Galilei

Green Software: The Awareness Awakens

Author:
J.M.Broekman
Student ID: 4230213
Email: j.m.broekman@students.uu.nl

First Supervisor:
dr. ir. J.M.E.M. van der Werf
j.m.e.m.vanderwerf@uu.nl
Utrecht University

Second Supervisor:
dr. F. Dalpiaz
f.dalpiaz@uu.nl
Utrecht University

First External Supervisor:
Msc. E.A. Jagroep
e.a.jagroep@uu.nl
Centric & Utrecht University

Second External Supervisor:
R. van Vliet
rob.van.vliet@centric.eu
Centric

Management Summary

This master's thesis describes the execution of a multiple embedded case study at a large international software producing company (SPO). The aim or our research is the creation of awareness on the topic of Green Software among the stakeholders of a software product. Green Software is environmentally friendly software that helps improve the environment by consuming less energy to run or by assisting other things going green. Green software is being discussed by academia, but remains relatively unknown to the industry. Therefore we conducted this research to create awareness among the stakeholders of a product software in order to introduce the concept to industry.

Our main research question is: How can we create awareness on the energy consumption of product software among stakeholders during product software development?

The main research question was supported by three sub-research questions:

1. How can we measure awareness on software energy consumption?
2. How can we trigger awareness on software energy consumption among stakeholders of a software product?
3. How to integrate the stimulus in a software development environment?

After a preliminary literature study on green software and awareness, we concluded that the best strategy is to create an energy dashboard and present it to the stakeholders in order to make them aware of the software energy consumption. The energy dashboard consists of a radar chart with performance metrics and the software energy consumption among the axes and introduced the concept of the Resource Utilization Score (RUS). The RUS is the surface of the figure on the radar chart and could be considered as the footprint of the software product. The RUS allows for comparison among different releases of the same software product and supports communication on the resource utilization by the software product. We created two surveys besides the energy dashboard. The first survey quantifies awareness on the software energy consumption among stakeholders of the software product. The second survey quantifies the acceptance of the energy dashboard among the stakeholders of the software product. We conducted

a multiple embedded case study among the two software products DocGen and RetailSystem. DocGen is a document generator, with a development team of 5 people in The Netherlands. RetailSystem is a transaction system for cash registers, with a development team of around 25 people spread across Belgium and Romania. Each case lasted for four development sprints, with each sprint lasting 3 weeks. We presented our dashboard during the sprint review meetings at the end of each sprint. During the presentation we asked them to fill in our surveys followed by a short presentation of the results of the delivered version compared to the previous version.

Our results indicate that we did create awareness on software energy consumption among stakeholders of the software product, although not always with positive scores. A closer look at the results revealed a knowledge gap among the participants. The participants find it difficult to address the software energy consumption and therefore it is required to create a knowledge bank. The development teams of both cases, accept the energy dashboard but do not always consider it an useful addition to their work. However, they do find it interesting and find the energy dashboard easy to use. Finally, the participants from both cases are willing to address the software energy consumption, if it is requested by the company or the customer.

In conclusion, we have created awareness on the energy consumption of product software among the development teams by introducing the energy dashboard at the sprint review meetings. Based on our results, we propose four green initiatives to improve the acceptance of the software energy consumption.

1. Identify all of the other stakeholders and their information needs.
2. Integrate the energy dashboard into the development process and automate the measurements.
3. Create a knowledge bank to address the knowledge gap of the developers.
4. Go public within the company, and embed the energy dashboard into SonarCube.

Contents

Chapter 1

Introduction

Sustainability and pro-environmental behavior has become an important topic in today's society. Climate changes, depletion of natural resources and high energy consumption made us aware of our impact on the environment. The role of Information Technology (IT) in this situation is twofold as it has the potential to identify energy savings but at the same time requires significant resources.

Attempts to reduce the resources of IT mainly focused on reducing the energy consumption of hardware as it is one of the most effective strategies to reduce energy consumption (Kazandjieva, Heller, Gnawali, Hofer, & Kozyrakis, 2011). The IT sector followed this strategy for many years by creating smaller and energy efficient chips. For example the energy requirements of a transistor on a chip has been relatively reduced by a million times compared to thirty years ago. (Preist & Shabajee, 2010).

As hardware became more energy efficient, the role of software gained little attention. However, software invokes hardware to execute instructions and therefore has a significant role in the energy consumption of IT (Noureddine, Rouvoy, & Seinturier, 2015). The current IT services account for 2% of the world's power capacity and shall increase to 18% in 2030 (Preist & Shabajee, 2010). To address these issues, a sustainability perspective on IT was introduced by academia. This perspective comes with different terms such as *Green IT* (Murugesan, 2008), *Green Software* (Lago et al., 2013) and *Sustainable Software* (Dick & Naumann, 2010). In this study we apply the term *green software*, related to the software energy consumption (SEC).

To create green software, the sustainability aspects need to be addressed during the early development stages and monitored during the software product lifecycle (Dick & Naumann, 2010). The benefits of green software are best realized with the development of product software as the effects of optimizing the product are multiplied by the number of installs. Therefore, small optimization's can lead to significant energy reductions.

To introduce this perspective in software engineering, stakeholders (e.g. the development team) of software need to become aware of the energy consumed by their software product. We specifically target the development teams as they are responsible for the creation of the software. Awareness on green software is currently lacking among stakeholders involved with the development of software in Software Producing Organizations (SPO) (Becker et al., 2015; Pang, Hindle, Adams, & Hassan, 2015). **Problem statement** - Therefore, we need to investigate how we can create awareness on the topic of green software among the stakeholders involved with the development of product software.

Our study focuses around the development teams of two software products of an international SPO. The next chapters describe the execution of this study on the creation of awareness on the energy consumption of software among the development teams. Chapter two presents the research approach with the research questions and the research method. Chapter three describes the background of green software in the field of software engineering. Chapter four describes the background of the concept awareness. Chapter five presents the research design about the general execution of our research. Chapter six presents the results first case around a software product named 'DocGen'. Chapter seven presents the results of the second case around a software product named 'RetailSystem'. Chapter eight contains the discussion around the results from both cases. Finally chapter nine presents the conclusions by looking back at the research questions.

Chapter 2

Research Approach

This chapter elaborates on the subject of creating awareness on the energy consumption of software among the development teams, by defining the relevance of the perspective of society and scientific. Based on the problem statement we define a set of research questions to set a scope for this research. This chapter ends with the description of the research method.

2.1 Relevance

This section describes the relevance of green software from a societal and scientific perspective.

2.1.1 Societal

IT is widespread throughout our society. Nowadays, almost every citizen in The Netherlands is in possession of a smartphone and the hours of on-demand video streaming almost surpass the hours of regular tv. All of these IT innovations require significant energy resources. Future estimates range from 14,5% of the global energy consumption in 2020 (Vereecken, Van Heddeghem, Colle, Pickavet, & Demeester, 2010) to 18% in 2030 (Preist & Shabajee, 2010). Society has become aware of its high energy consumption and invents countermeasures focusing on the energy efficiency of hardware such as energy labels for consumer electronics.

Another example is the growing demand for sustainable services from companies. A significant amount of companies commit themselves to Corporate Social Responsibility

(CSR (MVO in Dutch)) treaty in The Netherlands. The CSR is an organization which promotes transparent sustainable development among corporations and government based on the ISO 26000 – Social Responsibility. Companies and governmental agencies demand commitment to these treaties as a condition for delivering products and services to their organization. Investing and promoting sustainable software would be a large contribution to the sustainable development of a SPO. At the same time it would add a competitive advantage to the position of the SPO(Porter & Van der Linde, 1996).

2.1.2 Scientific

In the last years, research on green software has emerged with the start of conferences on sustainability and software such as the ICT4S conference. But also the increase in studies about measuring the energy consumption of software in detail (Bozzelli, Gu, & Lago, 2013). Currently we are at a point of introducing the concepts of green software into practice. This introduces a new research area of creating awareness among stakeholders of software to embrace the concepts of green software. Recently the first preliminary researches on awareness about green software appeared (Manotas et al., 2016). However, the number of studies about the adoption of green software are rare.

2.2 Research Questions

The main research question is introduced in this section. By answering this question, together with the sub-questions, a basis is developed to investigate awareness on energy consumption of product software among stakeholders in a software development environment. The main research question is:

RQ: How can we create awareness on the energy consumption of product software among stakeholders during product software development?

The main research question is supported by three sub-questions:

SQ1: How can we measure awareness on software energy consumption?

Awareness is a broad concept applied among many fields of science. As it is an odd concept in the field of software engineering, our study requires a thorough literature study among many fields of science, to investigation this concept.

SQ2: How can we trigger awareness on software energy consumption among stakeholders of a software product?

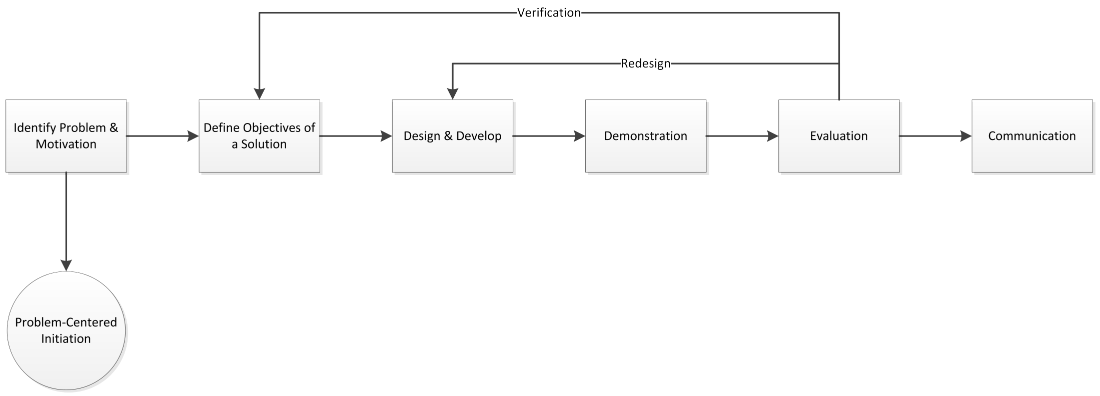

Verification
Redesign
Identify Problem & Motivation
Define Objectives of a Solution
Design & Develop
Demonstration
Evaluation
Communication
Problem-Centered Initiation

Figure 2.1: Design Science Research Method derived from (Peffers et al., 2007)

Once we have defined the concept of awareness we need to be able to apply it in order to trigger awareness about the energy consumption of software. Triggering awareness can be derived from literature, but there are also many examples from practice which might provide clues to trigger awareness.

SQ3: How to integrate the stimulus in a software development environment?

With the result of SQ2 we hope to be able to trigger awareness among the stakeholders. However, triggering the awareness is not enough. In order make full use of the potential of green software we need to know how we can maintain this awareness and keep the attention of the stakeholders. Therefore, we need to define an approach to integrate the stimulus in the software development environment to maintain the awareness on software energy consumption.

2.3 Method

The applied research method is a mixed research method with both qualitative and quantative methods. The main research method is the Design Science Research Methodology (DSRM) for information systems (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007). A literature study, multiple embedded case study and a survey were performed within the six phases of the DSRM (see fig 2.1). The next section present the different research methods and the rationale.

2.3.1 Design Science

The main research method is design science which creates and evaluates IT artifacts intended to solve identified organizational problems (von Alan, March, Park, & Ram, 2004). Within the design science method we follow the guidelines of the Design Science Research Methodology (DSRM) for information systems research (Peffers et al., 2007). The method consists of six stages, each with its own activities. For this study we followed all six stages. The research method starts at the first stage, as the research has a problem-centered initiation. The evaluation phase supports the idea of improving the artifact by returning to the design & develop phase. It also encourage the researchers to compare the results from the evaluation stage with the previous defined objectives in order to check whether goals are achieved. Finally, it concludes the method with the communication phase to communicate the results to the stakeholders of the project. These stakeholders are the supervisors of the project who have an interest in the results.

2.3.2 Literature Study

To identify the problem we start with a literature study. The literature study focuses on gathering information about the topics covering green software. Literature is gathered via Google Scholar from different publishers by applying the guidelines of *snowballing* (Wohlin, 2014). Sustainability transcends multiple disciplines (Becker et al., 2015), this directs this research also to other fields of science.

2.3.3 Multiple Embedded Case Study

A case study is applied when the boundary between the phenomenon and its context may be unclear (Yin, 2013). There are two units of analysis, the awareness of the stakeholders on the software energy consumption and the acceptance of the trigger to create awareness. The second unit of analysis is required to determine whether we chose the right trigger to introduce the concept of green software. Both units of analysis are investigated at two separate software product of an international software producing organization. The two separate software products with the two units of analysis qualify as a multiple embedded case study. The layout of the embedded case study is graphically shown in fig 2.2.

Each case within the multiple embedded case study lasted for four development sprints. Every sprint lasted 3 weeks and after every two sprints, a new release was delivered. To keep triggering the awareness of the stakeholders during the non-deliverable sprints, we included nightly builds to present a new dashboard during every sprint review meeting.

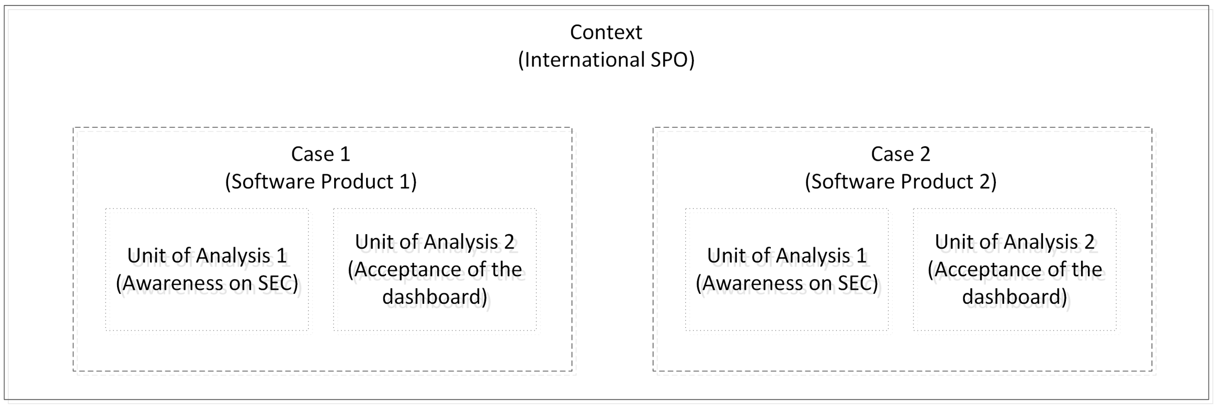


Figure 2.2: Layout of the embedded case study derived from (Yin, 2013)

2.3.4 Survey

A survey was conducted to quantativly evaluate the influence of the artifact. The questionnaires within the survey consist of Likert-Type data-items, instead of Likert-Scale, due to a lack of a test population to validate the items statistically. The Likert-Type data items also limit the statistical tests to the non-parametric spectrum (Boone & Boone, 2012).

Chapter 3

Background: SEC in Software Engineering

The next sections describe the role of the software energy consumption in software engineering. The chapter starts with the introduction of energy efficiency in computing, followed by the background literature on green software and the role of product software.

3.1 Energy Efficiency in Computing

In the past decades the computational power has grown significantly. One of the most influential studies in this field is conducted by Moore (1965) which states that the number of transistors in integrated circuit doubles every two year. This observation was later coined as Moore's law. An overview of this development is presented in figure 3.1. An addition to this field of development is proposed by Koomey, Berard, Sanchez, and Wong (2011) with the observation of the computations per joule of energy in computing hardware. According to Koomey et al. the number of computations per joule of energy doubles every 1.57 years since the 1950s till the present as can be seen in figure 3.2. The energy efficiency improvements introduced the start of the mobile devices as we now use on daily base.

Based on the two theories presented in the previous paragraph we can conclude that the energy efficiency and performance of hardware has made significant improvements. However, as hardware became faster, the software became slower. One of these observations was stated by Wirth (1995) and resulted in Wirth's Law: software is getting slower more rapidly than hardware becomes faster. This development is acknowledged by industry pioneers such as Bill Gates with the comment: 'The speed of commercial software

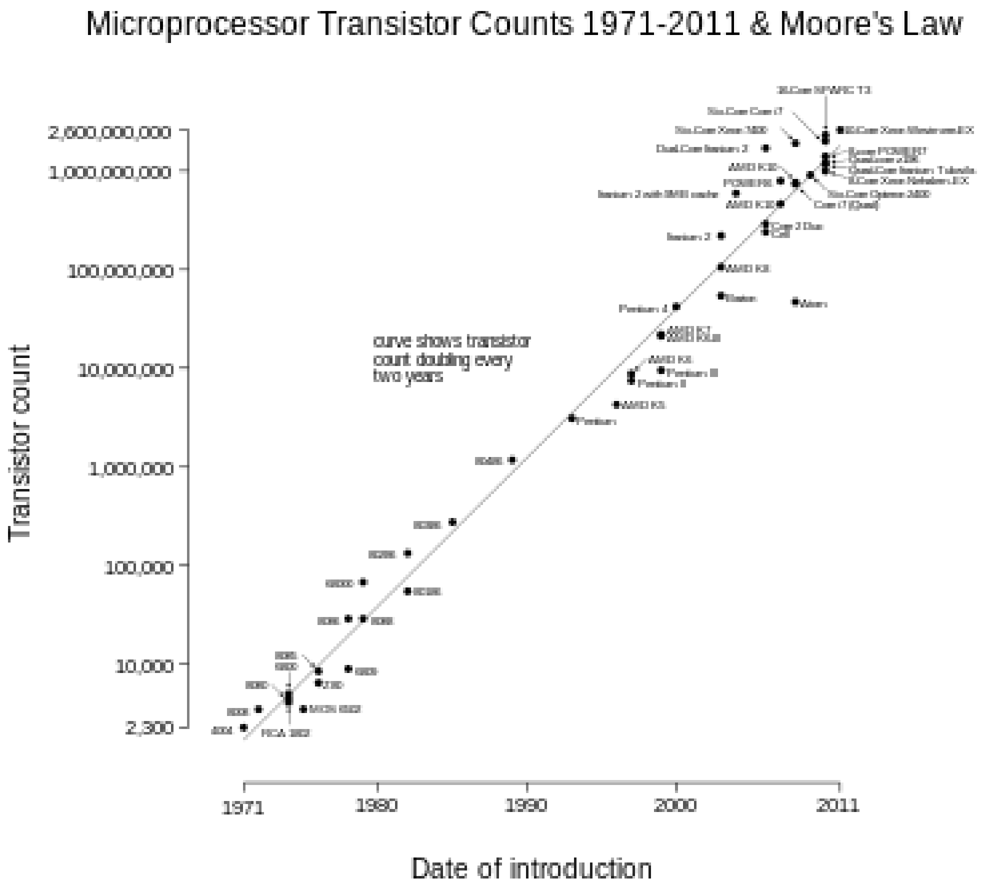


Figure 3.1: Microprocessor Transistor Counts 1971-2011 & Moore's Law, derived from (Commons, 2011)

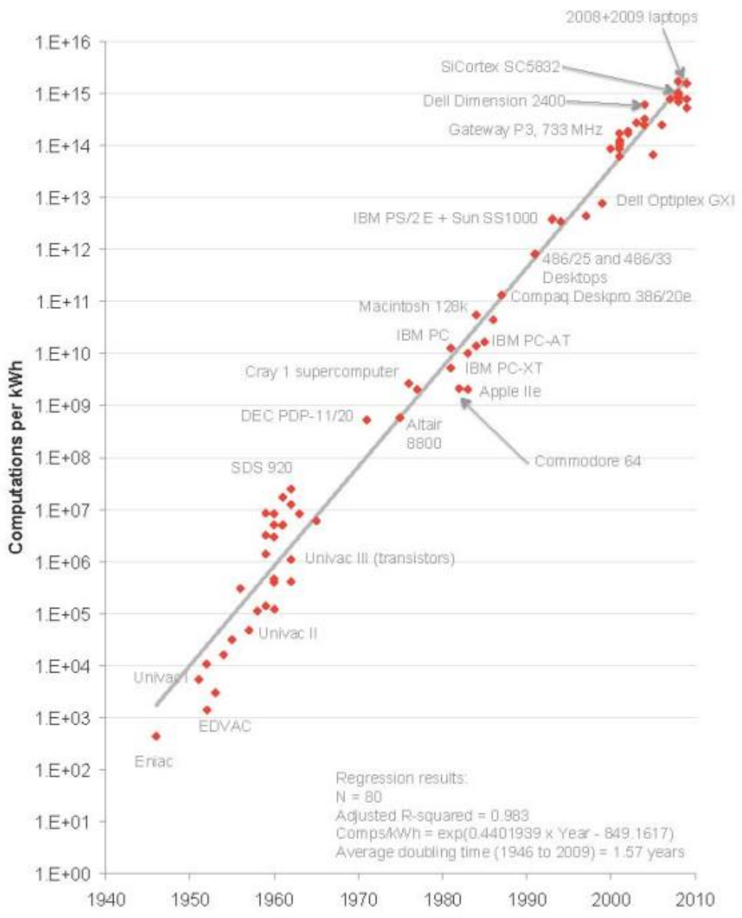


Figure 3.2: Computations per kWh, from 1946 to 2009, derived from (Koomey, 2009)

generally slows by 50% every 18 months, thereby negating all the benefits of Moore's law'.

3.2 Green Software

Sustainability is generally defined as 'the capacity to endure' (Lago, Koçak, Crnkovic, & Penzenstadler, 2015) and consists of five dimensions *social*, *environmental*, *economical* Brundtland et al. (1987), *individual* (Goodland, 2002) and *technical* (Penzenstadler & Femmer, 2013). The field of green software can be related to the environmental dimension of sustainability as it is defined as: 'Environmentally friendly software that helps improve the environment by consuming less energy to run or by assisting other things going green. It is also features characteristics that make its code or modules reusable' (Murugesan, 2008).

Energy efficiency or green software is an important topic in the field of mobile development. A high resource utilization drains the battery and cause frustration among end-users. Therefore, energy efficiency is an acknowledged software quality for mobile applications (apps) as it is something tangible. For traditional software engineering it is not a software quality, as the availability of energy is not an issue. There are limitations to the energy consumption, but this only becomes a limitation with large scale hardware projects, such as datacenters. Therefore, energy efficiency is not a software quality with 'traditional' software engineering targeting non-mobile devices. To change this into a tangible concept, we need to measure the energy consumption of software. Measurements are hard to conduct due to limitations of the measurement techniques and virtual machines sharing hardware (Seo, Malek, & Medvidovic, 2007; Kansal, Zhao, Liu, Kothari, & Bhattacharya, 2010; E. A. Jagroep et al., 2015).

The results provide valuable insights about energy hotspots on system and component level of the software architecture. Previous work by E. A. Jagroep et al. (2015) used this approach to reduce the energy consumption of a software product by 67%. Based on this case, the authors proposed an energy perspective on software architecture to identify energy hot spots and reduce the hardware utilization (E. Jagroep, van der Werf, Brinkkemper, Blom, & van Vliet, 2016).

A similar approach to reduce the energy consumption is to compare different versions of a software product to relate energy consumption to code changes. (Hindle, 2012) investigated code changes among five hundred daily builds of Firefox 3.6. The results showed little variance among the releases as the stability of the browser increased. Another study about energy consumption and internet browsers by Rasmussen, Wilson, and Hindle (2014), investigated the role of advertisement blocker plugins (adblockers) to increase the battery duration of smartphones. The results showed little increases in battery savings, but only in specific use cases. Still, small optimization's among worldwide used applications can make significant reductions.

A common mistake about green software is the trade-off between energy consumption and functionality. Some software product require significant hardware resources to execute the functionality of the software. For example, a business intelligence (BI) reporting tool requires more resources than a static web page. Green software does not exclude hardware intensive software, but it asses the functionality in the relation to the energy usage. A query in the BI tool might cost a lot of energy, but the energy usage might be reduced by applying caching or optimizing the query. Therefore green software searches for the most energy efficient way to execute the functionality of the software instead of excluding the functionality.

3.3 Product Software

Product Software is defined as a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in specific market (Xu & Brinkkemper, 2007). The main focus of product software is on the development of the product itself, opposed to tailor-made software which focuses on the demands of a specific customer. A software product is released among several customers. Small optimization's allow for large contributions as the benefits are multiplied by the number of installs. Therefore, the benefits of green software are best realized with the development of product software.

Product Managers are responsible for the functionality of the product and collaborate with several internal and external stakeholders (Xu & Brinkkemper, 2007). Bekkers, van de Weerd, Brinkkemper, and Mahieu (2008) define the following stakeholders of a software product.

Internal:

- The Company Board
- Research & Innovation
- Consultants of the services
- Development team(s)
- Support
- Sales & Marketing

External:

- The Market
- Partners
- Customers

Both the internal and external stakeholders have an interest in the sustainability aspect of the software product. The company board is responsible for the mission, vision and strategy. In the Netherlands, more companies become aware of their impact on the environment and introduce green strategies to align business with sustainability. Research and innovation explores new opportunities in order to address sustainability issues in their products and service (e.g. new technologies or energy usage). Development is responsible for developing the software product itself. The choice of technology, architecture, design patterns and methods influence the sustainability of the software product. The most important external stakeholder regarding sustainability is the customer. The customer uses the software product and has the power to demand a sustainable product as it is more and more expected by their own customers.

In this project we target Development as the most important stakeholder as it is responsible for implementing sustainable practices into the software product.

Chapter 4

Background: Awareness

This chapter introduces the background theory of the concept *awareness* and is based on three main sections. The first section is about defining awareness and how it is applied in the field of green software. The second section continues with background theories of creating awareness. The third section concludes with how we can measure awareness.

4.1 Defining Awareness

A problem with the concept awareness is the lack of an uniform definition. Different fields of science such as psychology and neuro-sciences has different views on the concept and therefore an uniform definition is lacking. In The International Dictionary of Psychology the concept awareness is synonym to the term *consciousness*, which is defined as 'the having of perceptions, thoughts and feelings; awareness' with the addition of 'the term is impossible to define except in terms that are unintelligible without a grasp of what consciousness means' and concludes with 'Nothing worth reading has been written about' Sutherland (1996). According to Sutherland it is impossible to define the term as we still are unable to understand it completely. Therefore, the coined definition is still vague and hard to apply. As an alternative we use the definition of the Cambridge Dictionary, which is 'knowledge that something exists, or understanding of a situation or subject at the present time based on information or experience'. In the context of green software we apply it as: 'understanding of the subject of software energy consumption at the present time based on information or experience'.

4.1.1 Awareness on Green Software

Hitherto, there are already studies towards awareness on green software. Lago and Jansen (2010) describe three awareness areas about green software in the context of Service Based Applications. The three areas are *process awareness*, *people awareness* and *service awareness*. *Process awareness* focuses on the development process of software. The problem of this area is to raise awareness of the impact of the software on the environment among decision makers. *People awareness* targets the awareness of the end-users of the software. *Service awareness* focuses on the reduction of the energy consumption of the service itself. All of the three aspects are measured by green metrics. The focus of this project is on process- and service awareness because we target stakeholders involved in the development process in order to create awareness about the energy consumption of the software product. The green metrics provided for process- and service awareness are related to the execution of the code and organizational green metrics such as printing budgets and not to the perception of an individual.

Pang et al. (2015) did a survey among hundred developers to find out what programmers know about software energy consumption. The main conclusion is a lack of knowledge among the developers and a low prioritization by customers.

In conclusion, there are studies about awareness on green software, but each study applies a different view on the concept of awareness. A uniform view or definition of awareness on green software is lacking. In this study, we define awareness as:

4.2 Creating Awareness

This section elaborates on how we can create awareness on the energy consumption of software based on existing theories. Raising awareness to reduce energy consumption has been widely applied among households in the past four decades. In the last decade, the use of dashboards to provide real-time feedback has proven to be the most successful approach (Hazas, Friday, & Scott, 2011). Especially, when users were able to compare themselves with peers. Therefore, a dashboard which provides feedback on utilization metrics of the software product could be a successful approach to create awareness on the energy consumption of software.

4.2.1 Eco-Visualizations

Raising awareness about energy consumption is investigated by the field of *Eco-Visualizations* (EV) and is defined as 'the real time consumption statistics of key environmental resources for the goal of promoting ecological literacy (Holmes, 2007). In addition, Pierce, Odom, and Blevis (2008) proposed EV as 'any kind of interactive device targeted at revealing energy use in order to promote sustainable behaviors' or foster positive attitudes towards sustainable practices.' This definition is applicable to this project as it matches the same goal, although the focus of this study is on software and not on a device.

Pierce et al. (2008) describe several aspects of eco-visualizations and their applicability. The aspects are: *feedback types*, *use-contexts* and *strategies* for designing effective EVs. *Feedback types* determine the data and type of visualization. The data refers to the scale (size) of the measurement while the visualization is the expression. Visualizing the energy consumption of a particular building consist of different data than the energy consumption of a city. The visualization also determines the feedback provided to the user. There is a trade-off between the artistic visualization and the pragmatic visualization. The first one refers to the visuals itself and should be 'present enough of an enigma to keep an audience interested without being easy to solve'. The pragmatic refers to the understanding of the data and should be 'designed to remove any sublimity, and instead foster immediate understanding'.

Use-Contexts refers to the trade-off between the influence of the person and a third-party on the behavior. The paper provides examples of this trade-off by focusing on energy consumption in different settings. In a home environment the owner has full control on the energy consumption of its house. The opposite would be in an office environment with regulated heating and light controlled by a third-party. A similar trade-off occurs within the field of green software, as it is the question who is in charge of the energy consumption of software. On the one end there is the developer who writes software which utilize the hardware and consumes energy. On the other end, there are specifications and software qualities (security, performance, availability, etc.) determined by a third-party (software architect, client or regulations) which also influences the energy consumption.

The third aspect is strategy and refers to different strategies which are related to different goals. The eight proposed strategies are:

1. Offering behavioral cues and indicators.
2. Providing tools for analysis.
3. Creating social incentive to conserve.
4. Connecting behavior to material impacts of consumption.
5. Encourage playful engagement and exploration with energy.
6. Projecting and cultivating sustainable lifestyles and values.
7. Raising public awareness and facilitating discussion.

8. Stimulating critical reflection.

The goal of this study is to raise awareness on green software among the stakeholders which aligns with strategy seven of the eight proposed strategies of EVs. By combining the work of (Pierce et al., 2008) on EV's with the results from (Hazas et al., 2011) we conclude that the creation of a dashboard to raise public awareness and facilitate discussion would be the best approach, to increases the awareness on green software among stakeholders of the software product.

The strategies proposed by Pierce et al. (2008) mention the use of social incentives and playful engagement. In addition, Pierce et al. state: 'Dormitories, apartments, office buildings and other use-contexts with high third-party control offer strong opportunities to explore competition as an incentive'. The combination of these findings might indicate the application of *Gamification*. Gamification can be defined as: "the use of game design elements in non-game contexts" (Deterding, Dixon, Khaled, & Nacke, 2011). In a follow-up, the author looked back at how gamification was applied in the past few years (Deterding, 2014). One his conclusions is the wrongfully application of the term 'Gamification'. In practice, the term has been used to solve the problems such as leaving customers by introducing 'loyalty points' or make work more exciting by using leaderboards to stimulate competition between employees. According to Deterding, these bad examples do not match the concept of gamification and fail in the end as they do not target the source of the problem. Therefore, applying gamification might be an option but it requires a thorough investigation of how game design elements can be applied in the working environment of development teams to create and maintain awareness on green software.

4.3 Barriers to reduce energy consumption

The previous literature provided many starting-points to reduce the energy consumption. To fully understand the adoption process of reducing energy consumption, one has to understand the barriers as well. The main difference between most research on energy consumption and this research, is the setting. Most research focus on consumer environments instead of a company environment. In a consumer environment the owner is responsible for the energy usage and pays the bill. In a company environment the energy user does'not pay the bill which introduces the *Principal-Agent problem* (PA-problem). The PA-problem addresses the conflict of interest between two parties of a contract. For example, a developer writes inefficient code which utilize extra hardware and thus energy. The company wants to reduce the energy costs by forcing the developer to write more efficient code. However the developer does not share this interest because it introduces more complexity to his workload. Graus and Worrell (2008) provide a table for the Principal-

Table 4.1: Categorization of the PA-problem derived from (Graus & Worrell, 2008)

	Chooses Technology	Does Not Choose Technology
Pays Energy Bill	**Cat 1**: potentially no PA-Problem	**Cat 2**: Efficiency Problem
Does Not pay Energy Bill	**Cat 3**: Usage and efficiency problem	**Cat 4**: Usage Problem

Agent classification of energy end users (Fig. 4.1). In the case of software engineering, a Cat. 3 PA-problem is introduced. The developer does not pay the energy bill and chooses the technology. The latter could be argued since the chosen technology is determined by many factors but the implementation (code instructions) of the specific technology is still decided by the developer. Thus, a usage and efficiency problem arises.

Another effect to take into consideration is the *Rebound Effect*, also known as Jevons paradox. The rebound effect was firstly described by Williams Stanley Jevons during the industrial revolution (Jevons, 1906). A new design of the steam engine improved the efficiency and was expected to reduce the consumption of coal. However, the lower demand of coal caused a decrease in price and decreased the total cost of ownership of the steam engine. This introduced the steam engine to other industries which resulted in a rise for the demand of coal and an increase in the coal consumption. Instead of the expected decrease of the coal consumption, the opposite occurred. The same pitfall occures in household environments. Despite many awareness campaigns the energy consumption keeps growing over the past forty years (Hazas et al., 2011).

4.4 Measuring Awareness

Another issue with the concept of awareness is the inability to quantify it. In order to quantify a concept it needs to have a scale. The construction of such a scale is hard, or even impossible to realize as we have to understand the individual persons thoughts and emotions. According to Nagel (1974) this is impossible because we are not able to objectively describe someone else experiences, thoughts and emotions as we always observe it from our own perspective. Therefore, we can not measure awareness directly, but there are means to measure it indirectly. Measuring awareness indirectly is broadly applied by marketeers to measure brand awareness. More concrete, they measure the perception of people with Likert scales applied in surveys.

4.4.1 Environmental Education

When we searched for a survey on measuring awareness on sustainability and software, we found the sub-discipline of environmental education which encompasses the theories of teaching environmental related subjects. One of the studied subjects within this field is to measure the ecological worldview. The theories from this field can be applied to measure the opinions from our stakeholders on green software. There are three widely used measurement techniques to assess the ecological view of different target groups. These are the Ecology Scale (Maloney & Ward, 1973), the Environmental Concern Scale (Weigel & Weigel, 1978) and the New Environmental Paradigm Scale (NEP Scale) proposed by Dunlap and Van Liere (1978). However, the first two are dated by focusing on specific content related to the environmental context of that time. To improve the NEP scale, a revised version of it has been proposed and validated Dunlap, Van Liere, Mertig, and Jones (2000). A similiar approach was conducted by Özden to measure the environmental awareness and attitude among Turkish student teachers. In this research a questionnaire was constructed based on Likert-scale items. The problem with all of these validated surveys are the worldwide ecological problems they address, which are hard to relate to the teachings of green software.

4.4.2 Environmental Psychology

A related sub-discipline is the field of environmental psychology. Within this field we found an interesting framework proposed by Matthies (2005). The framework describes the process of changing environmentally detrimental habits based on four different phases. The first phase is the *norm activation* which targets three different consciousness levels in relation to environmental problems. The second phase is the *motivation* which introduces the motivations to change behavior from the personal norm, the social norm and the cost of action. The third phase is the *evaluation* which is a weighted sum of all the previous aspects. Finally, the fourth phase is the action as result of the previous phase, either the modified or old behavior.

Measuring awareness is a difficult topic as it is something we can not measure directly. Instead, we can use a test to quantify the opinion of people regarding green software. The current tests which measure the ecological view of participants are too broad and can not be applied to the concept of green software. As an alternative we need to develop a new test to capture the view of the participants on the energy consumption of software. The model of 'changing environmental detrimental habits' is a promising starting point, as it clearly describes different aspects of norm activation and motivation of the participants.

Chapter 5

Research Design

This chapter describes the research design of the study. Section one starts with the development of an eco-visualization by means of an energy dashboard. Section two describes the experiment of measuring the energy consumption of the two software products. Section three describes both surveys to measure the awareness and to measure the user acceptance. Section four is the Case Study Protocol, which describes the execution of the case study.

5.1 Energy Dashboard

Following the theories of eco-visualizations as defined by Pierce et al. (2008), we designed an eco-visualization to communicate the sustainability aspects of a software product to the stakeholders in order to raise the awareness and facilitate discussion on green software.

As *feedback-type*, we chose a dashboard, as it is one of the most effective feedback-types to communicate energy consumption. As for the *data* represented by the feedback-type, we defined a set of performance metrics besides the energy consumption (E. A. Jagroep, van der Werf, Broekman, et al., 2016). The reason for including the performance metrics is to create a broader view on green software and include the environment (platform) of the software. Software requires hardware to execute the instructions. The functionality of the software influences the hardware utilization and thus the energy consumption. For example, a business intelligence application requires significant resources compared to a static web form. However, it is very hard to compare software functionality quantitatively. Therefore we analyze the software in relation to its environment by the performance aspects and use these aspects to calculate a score labeled as the Resource Utilization Score (RUS). The RUS cannot be applied across different software due to the different

Table 5.1: Metrics of the radar chart

Metric	Unit	Description
CPU	%	CPU utilization expressed by a percentage of the total
Memory	bytes	Memory utilization by the software
Hard Disk Utilization	%	The utilization of the hard disk expressed by percentage
Hard Disk Transfer	Bytes/s	The transfer rate of data
Network Transfer	Bytes/s	The sent and received data through the network controller
Execution Time	Time (s)	The execution time of a specific task
Energy	Joule (j)	The energy consumption to execute a specific task

nature of the products. However, it is possible to compare it with different versions of itself to identify performance issues. This approach aligns with the life cycle of product software which is continuously developed and improved. Every new release of the product can be compared with the former release to identify possible performance issues. As for the *visualization*, we chose a radar chart to present the metrics on the axes.

5.1.1 The metrics

We identified seven aspects to assess the software products (E. A. Jagroep, van der Werf, Broekman, et al., 2016). Table 5.1 provides an overview of the seven metrics. The selection of these seven aspects are based on discussions between the author, academia and professionals from the field of software engineering. The identified aspects are suitable for the stakeholders of development. The list of aspects can be adjusted by including or removing an aspect depending on the stakeholder. By no means represent these seven aspects a complete list of software assessment.

5.1.2 The Radar Chart

The axes of the radar chart represent the metrics of the software product. Each of these metrics has a different scale (i.e. percentage, bytes or bytes/sec) which makes it difficult to create a graph with a uniform scale. However, our focus is the differences between the releases and therefore we are able to use the relative differences of the same metric between two different versions. The relative difference is expressed by a percentage which transforms all of the metrics to one scale, as required for radar charts (Schütz, Speckesser, Schmid, et al., 1998).

A decline in relative difference results in a negative number. Negative numbers are hard to display in a radar chart as it would represent a value on the other side of the center on the same axis, this would result in unreadable figures.

Analysis of preliminary data provided us with valuable information. The minimum relative difference is -100%, as a value can not be less than zero. The maximum relative difference is theoretically infinite as values may increase significantly. To set a boundary for the graphs we limit the range from -100% to +100%. An increase of more than 100% is possible but cut off at the borders to keep the readability of the graph. The result is a radar chart with a range from -100 till +100 instead of a range from 0 to 1 (or 100).

The center of the range (0%) requires special interpretation in the comparison. An increase (or decrease) of 0% indicates no change compared to the previous version. Therefore a line is drawn around the 0% to indicate the performance of the previous version.

The relative difference is calculated by subtracting the new value from the old value, divided by the old value and multiplied by 100%. The formula is provided in calculation 5.1 and an example of a change in CPU utilization is provided in 5.2.

$$Metric\,Change = \frac{(newValue - oldValue)}{oldValue} \cdot 100 \tag{5.1}$$

$$CPU\ utilization\ change = \frac{(60\% - 45\%)}{45\%} \cdot 100\% = 33.33\% \tag{5.2}$$

The figure of the radar chart is filled with a heat map. In the heat map green symbolizes a decrease, yellow a slight increase, orange for a moderate increase and red for a big increase.

The result of all the previous design decision can be seen in figure 5.1.

5.1.3 Resource Utilization Score

During the development of the dashboard we searched for a method to create a footprint of the software product regarding its hardware utilization and energy consumption. After several brainstorm sessions we developed a radar chart with all of the assessed performance metrics on the axes. The surface of the figure (see fig 5.2) represents the footprint of the software product regarding the hardware utilization and energy consumption and was defined as the *Resource Utilization Score* (RUS).

The development of the RUS was based on the work of Schütz et al. (1998) which describe the four main goals of the radar chart:

1. Visualization of interrelated performance measures through standardized scales.

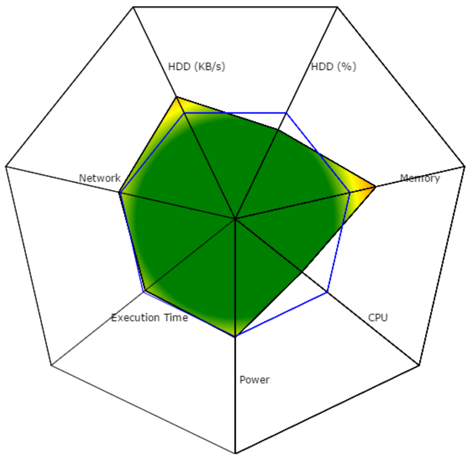


Figure 5.1: The applied radar chart

2. Produce an effective and revealing description of selective performance dimensions in just one synthetic indicator.
3. The change in overall performance between two points of time can be analyzed by comparing the difference in surface of the same object.
4. The shape of the surface allows for analysis of comparison between different objects.

Mosley and Mayer (1998) developed the *Surface Measure of Overall Performance* (SMOP) to calculate the surface of the radar chart. In this formula (5.3), the Px-values are the axes of the chart (see fig 5.2. The successive Px-values are multiplied and summed around all of the axes of the chart. This value is multiplied by the sine of pi divided by the number of axes (n).

$$SMOP = ((P1 \cdot P2) + (P2 \cdot P3) + (P3 \cdot P4) + ... + (Pn \cdot P1)) \cdot \sin(\frac{\pi}{n}) \tag{5.3}$$

However, formula 5.3 neglects the order of the axes. A different order of axes results in a different pairing of the axes and thus a different surface. Mosley and Mayer (1998) illustrate this problem and provide an example in which the SMOP can be three times as big based on a different order. To solve this problem, they calculate the average surface based on all possible orders of the axes. Instead of calculating for each possible order the corresponding surface, we calculate the triangle between every pair of axes P_i and P_j for an equal number of times. The approach results in formula 5.4:

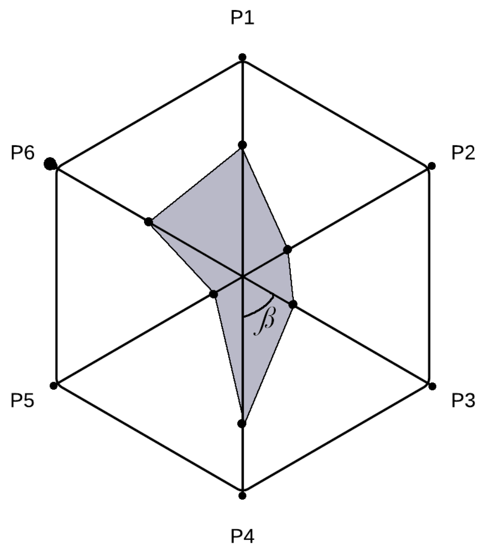


Figure 5.2: A radar chart with Px axes.

$$f(n) = \sin\frac{\pi}{n}(\sum_{i=1}^{n}\sum_{j=1}^{n}(P_i \cdot P_j)) \tag{5.4}$$

Formula 5.4 can be simplified. The first part of the formula is a constant factor as long as the number of axes (n) remain identical. Therefore it is possible to remove the first part($\sin\frac{\pi}{n}$) as long as all of the figures have the same number of axes. Note that by removing the first part, the formula no longer represents the surface of the chart, but a score based on the relation between the axes. Another observation are the pairing of the axes. In the current formula, the axes can be paired with themselves and calculated twice, as $P_i \cdot P_j$ is identical to $P_j \cdot P_i$. Removing the constant and adjusting the pairing results in formula 5.5:

$$f(n) = C\,(\sum_{i=1}^{n}\sum_{i<j}^{n}(P_i \cdot P_j)) \tag{5.5}$$

$$C = \sin\frac{\pi}{n}$$

One of the assumptions of the rader chart is the equal weight of every axis. However, some scenario's put more emphasis on one of the axes. To include this scenario in the formula, a weight factor was introduced (5.6). The downside of this weight factor is the

Table 5.2: The applied Performance Monitor Indicators

Dimension	Performance Monitor Indicator
CPU	Process (% Processor Time)
Memory	Process (Working Set - Private)
HDD Utilization	Physical Disk (% Idle Time)
HDD Transfer	Physical Disk (Disk Bytes/sec)
Network	Network Interface (Bytes Total/sec)

violation of the fourth goal described by Schütz et al. (1998) as different weight factors results in different shapes which can not be used for comparison across different objects.

$$f(n) = C\,(\sum_{i=1}^{n}\sum_{i<j}^{n} W_{Pi,Pj}(P_i \cdot P_j)) \tag{5.6}$$

$$C = \sin\frac{\pi}{n}$$

$$W_{Pi,Pj} = \begin{cases} 1, & if P_i and P_j are adjacent axes \\ 0, & otherwise \end{cases}$$

5.2 Software Energy Consumption Measurement

To measure the energy consumption we used Joulemeter developed by Microsoft. Joulemeter is a software program that is able to measure the energy consumption of a process of the Windows Task Manager. Joulemeter needs to be calibrated on the machine it's installed. Calibration occurs by connecting a WattsUp pro energy meter to the machine. During the calibration Joulemeter builds a power model by stressing the hardware components of the machine and combines it with the energy values drawn from the energy meter. After this calibration the energy meter was no longer needed as the power model was saved to an XML file on the hard-disk.

The data collection of the performance aspects was performed by Performance Monitor, a build-in application for Windows. Performance monitor is able to gather data on hardware utilization of its host or of a remote machine. The program allows the user to define a custom set of performance counters to monitor the system. The selected performance indicators are presented in table 5.2. The collected data is stored in a .csv file.

Performance Monitor is able to collect data remotely or of its host. The remote approach collects the data from other machines and saves it to a log file on the logging

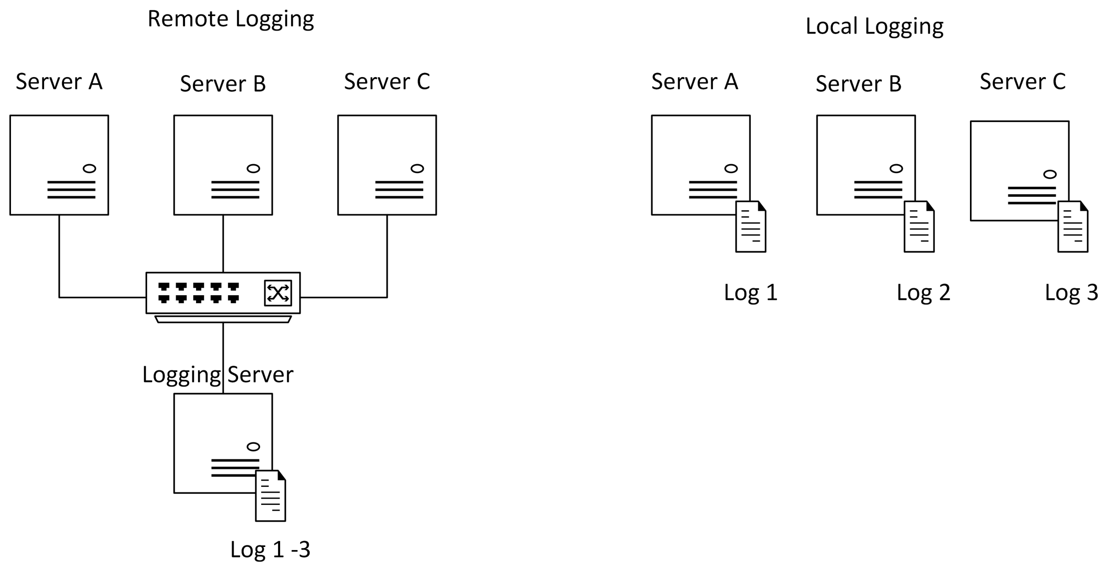


Figure 5.3: Example setup of logging scenario's

machine. This approach has minimal interference with the the resource utilization of the inspected machines, but increases the network activity. An example setup is presented on the left side in figure 5.3 with the logging server collecting data from server A, B and C via the network interface. The local approach collects the data of its host and saves it to a log file on the host machine. An example setup is presented on the right side in figure 5.3. The local approach causes more overhead on the host machine. Saving and collecting the data increases the memory and hard disk utilization of the monitored machine. Therefore, the remote logging approach is preferred to monitor the systems.

5.3 The Survey

The second main artifact is the survey. There are two surveys, one to measure the awareness on the SEC and one to measure the user acceptance.

Figure 5.4 provides a general overview of the construction of the two questionnaires. On the left side is the general theory which is applied and transformed into the specific output at the right side. The procedure can be read from left to right to follow the line of the construction and from right to left to trace back the design rationale. The first concept on the left side is the *Theoretical Model* which serves as input for the survey. The second concept is the set of *constructs* from the model. Each construct is measured by a set of *statements*. Finally the statements are transformed into directed (positive and

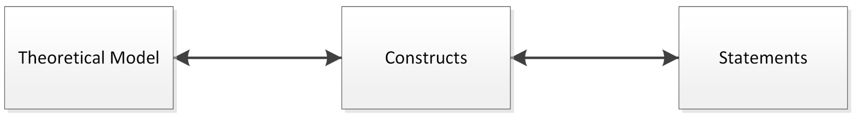


Figure 5.4: Survey construction overview

negative) statements to remove biased opinions regarding the sentiment.

5.3.1 Measuring awareness

The goal of the first survey is to capture the awareness of the stakeholder on the energy consumption of their software product. Hitherto, there is no uniform test or scale to measure this awareness. Özden (2008) performed a similar research by creating a survey to measure the opinions on sustainability awareness among Turkish students. However, the sustainability aspects of the survey are broadly defined and inapplicable for this survey. The statements of the survey are statistically validated with the data from 500 students. Unfortunately our study didn't have access to a population of this magnitude and therefore we are unable to validate the awareness survey statistically.

Theoretical Model

The first step in the process of creating the survey was to find a theoretical model which combines the concept of sustainability, human behavior and cognitive aspects. After an extensive application of snowballing during the literature study we found a model about changing environmental detrimental habits from the field of *environmental psychology* proposed by Matthies (2005). Figure 5.5 presents the model with the four stages: *Norm Activation*, *Motivation*, *Evaluation* and *Action*.

The Norm Activation is about becoming aware of environmental problems and the influence of one's behavior. It consists of three constructs: *Consciousness of the environmental problem*, *Consciousness of relevance of one's behavior* and *Consciousness of one's possibilities*.

Motivation refers to the opinions on pro-environmental behavior from different perspectives. The three constructs from the Norm Activation influence the *Personal Environmental Norm* which is the personal opinion on sustainability. Besides the personal motivation is the *Social Norm* which refers to the opinions of other persons in the direct environment of a person. Finally there is the *Other Motives* which describes the motivations unrelated to people. An example is the cost of the action to change the behavior.

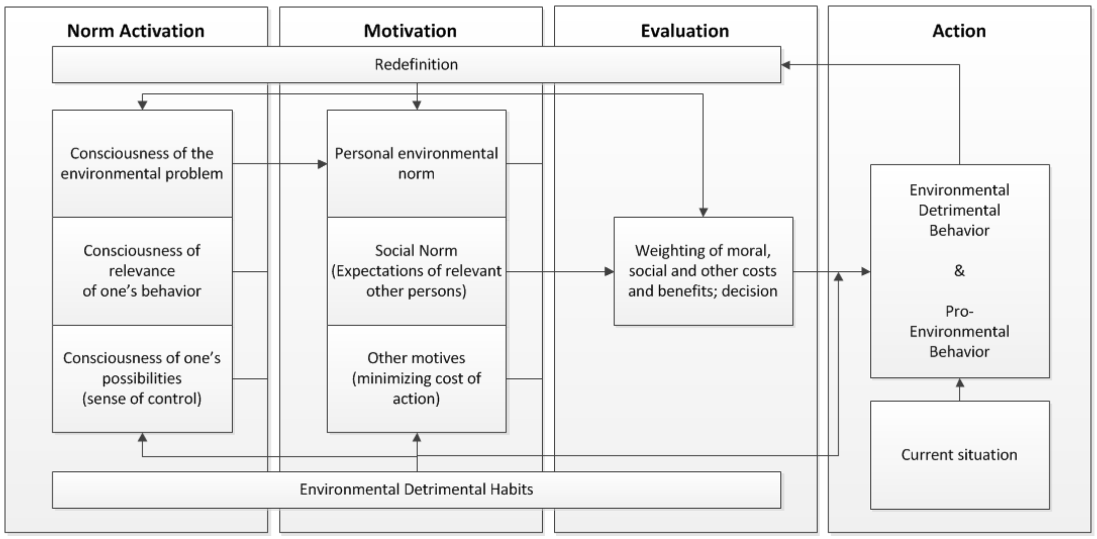

Norm Activation
Motivation
Evaluation
Action
Redefinition
Consciousness of the environmental problem
Consciousness of relevance of one's behavior
Consciousness of one's possibilities (sense of control)
Personal environmental norm
Social Norm (Expectations of relevant other persons)
Other motives (minimizing cost of action)
Weighting of moral, social and other costs and benefits; decision
Environmental Detrimental Behavior & Pro-Environmental Behavior
Current situation
Environmental Detrimental Habits

Figure 5.5: Model of changing environmentally detrimental habits derived from Matthies (2005)

.

After the norm activation and the motivation comes the evaluation which consists of weighting the benefits and costs of changing the behavior.

The final stage is the action as a result of the evaluation. When the benefits are more important than the costs, the person is able to change its behavior.

Constructs & Statements

Each stage of the model contains multiple constructs addressing a specific concept. To construct the survey we derive the constructs from the norm activation stage and the motivation stage. The construct from the evaluation stage is the result of the first two stages and can be classified as the dependent variable *awareness*. The construct from the *Action* stage is the action as a result of the awareness.

To summarize, there are six constructs derived from the first two stages of the model and transformed into constructs applied to measure the awareness on software energy consumption.

The first construct is *Consciousness of the environmental problem* and targets the general issues of environmental problems (climate changes, depletion of natural resources, etc.). This construct is transformed to target the general issue of sustainable software in the context of software development: *Consciousness of the energy consumption of software.*

Table 5.3: The transformed constructs

Code	Original Construct	Transformed Construct
C1	Consciousness of the environmental problem	Consciousness of the energy consumption of software
C2	Consciousness of relevance of one's behavior	Consciousness of relevance of creating energy efficient software
C3	Consciousness of one's possibilities (sense of control)	Consciousness of one's possibilities to reduce the SEC
C4	Personal opinion	Personal opinion on reducing the SEC during software development
C5	Social norm	Social norm on developing energy efficient software
C6	Other motives (minimizing cost of action)	Other motives to reduce the SEC

The second construct is *Consciousness of relevance of one's behavior* and measures if the person is aware of the influence of its own behavior regarding sustainability. In the context of our research this construct is transformed as: *Consciousness of relevance of creating energy efficient software.*

The third construct is *Consciousness of one's possibilities (sense of control).* This construct measures if someone is aware about its possibilities to change its behavior and become more sustainable. In the context of software engineering this construct is transformed into: *Consciousness of one's possibilities to reduce the SEC.*

The fourth construct is *Personal opinion* and adds extra reasons of changing detrimental habits. For example, a person can be: aware of climate change (con 1), know about its carbon footprint and energy usage (con 2), be a vegetarian (con 3), but still decide to go on holiday to the other side of the world by airplane twice a year because the person values exploring other cultures. Therefore the fourth construct is applied and transformed in: *Personal opinion on reducing the SEC during software development.*

The fifth construct is *Social norm* which represents the social environment of the person. The context of a company adds more complexity to the situation as different stakeholders have different interests. In the context of energy consumption in software development this construct is transformed into: *Social norm on developing energy efficient software.*

The sixth construct is *Other motives (minimizing cost of action)* and refers to the costs of changing the behavior. A practical example in the context of software engineering would be the cost of rewriting inefficient code. This construct is transformed into: *Other*

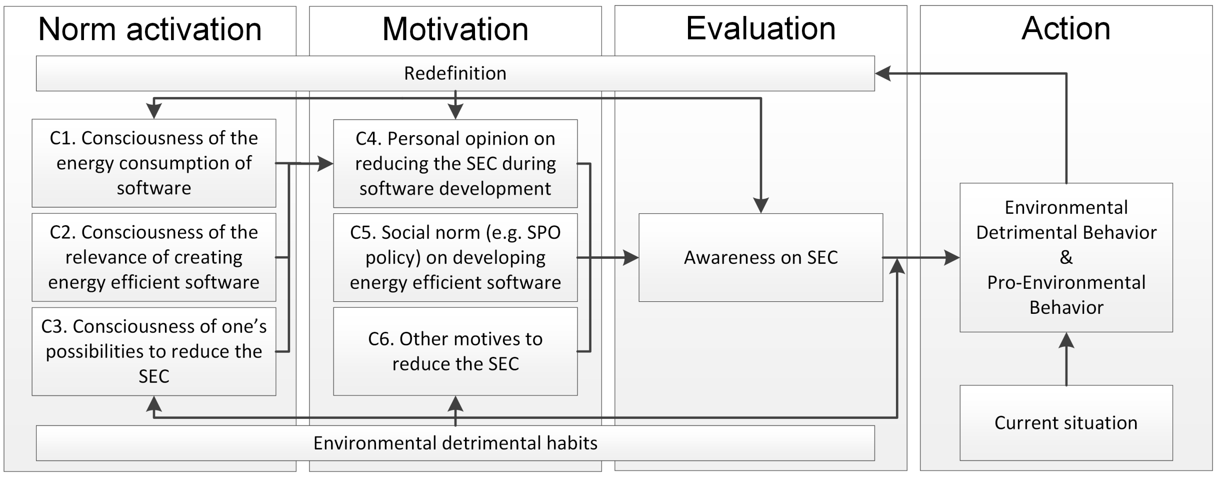

Norm activation
Motivation
Evaluation
Action
Redefinition
C1. Consciousness of the energy consumption of software
C2. Consciousness of the relevance of creating energy efficient software
C3. Consciousness of one's possibilities to reduce the SEC
C4. Personal opinion on reducing the SEC during software development
C5. Social norm (e.g. SPO policy) on developing energy efficient software
C6. Other motives to reduce the SEC
Awareness on SEC
Environmental Detrimental Behavior & Pro-Environmental Behavior
Current situation
Environmental detrimental habits

Figure 5.6: The transformed constructs in the original model inspired by Matthies (2005)

motives to reduce the SEC.

Statements

Each construct of the previous section is measured by statements. Each statement has a pro- and detrimental-environmental version of the same statement to filter inconsistencies and create a coherent opinion. The complete set consists of 29 statements (fourteen pairs and one individual statement). The tables 5.4, 5.5, 5.6, 5.7, 5.7, 5.8 and 5.9 present the statements of each construct. The first column contains the unique reference code of the statement consisting of the construct reference (Cx), the statement number and the direction with A (positive) or B (negative). For example, C1.1A refers to the positive statement 1 which belongs to construct 1. The second column presents the statement and the third column presents the order of the statement in the survey.

The statements were three times reviewed by different panels which led to three revisions of the statements. The first round was a group of 5 academics consisting of 3 PhD students and 2 Assistant Professors. The feedback mainly focused on the order of the questions and the applied language but they agreed with the idea of using positive and negative statements. After the feedback round, the language and order of the statements were adjusted.

The second round was among 6 IT professionals of 5 different companies with 2 to 36 years (average of 13) of experience. Their main concern was the order of the questions. The positive and negative statements were too close to each other which gave the idea of

Table 5.4: The statements of C1: Consciousness of the energy consumption of software

Code	Statement	Order
C1.1A	I want to determine the energy consumption of our software development environment (e.g. by calculating the number of (test) servers, laptops and other resources and their energy consumption.	1
C1.1B	Investigating the energy consumption of our development resources (laptops, desktops and (test) servers) in our software development environment is of no interest to me.	8
C1.2A	Before this project, I wondered multiple times about the energy consumption of our software development environment, but didn't do anything with it.	12
C1.2B	Before this project, I had never thought of the energy consumption of our software development environment.	5

Table 5.5: The statements of C2: Consciousness of the relevance of creating energy efficient software

Code	Statement	Order
C2.3A	I expect that software has a large influence on the energy usage.	9
C2.3B	I expect that the energy usage is only marginally influenced by software.	2
C2.4A	I would like to know the energy consumption of our software product.	6
C2.4B	The energy consumption of our software product is of no interest to me.	13

Table 5.6: The statements of C3: Consciousness of one's possibilities to reduce the SEC

Code	Statement	Order
C3.5A	If I had more time to work on the code, I would be able to reduce the energy consumption.	3
C3.5B	Code optimization to lower the energy consumption would be impossible despite extra time.	10
C3.6A	The applied techniques (programming language, architecture, design patterns, etc.) to realize the software product, allow for the reduction of energy consumption.	14
C3.6B	Energy consumption cannot be reduced due to the implemented techniques (programming language, architecture, design patterns, etc.).	7
C3.7A	It is possible to make a trade-off between our current software qualities (e.g. performance) and the energy consumption of our software.	11
C3.7B	Our software qualities (e.g. performance) do not allow room for implementing initiatives to address the energy consumption.	4
C3.8	The energy consumption should be addressed by: A) Developers B) Software Architects C) Developers and Software Architects D) Others, namely...	15

Table 5.7: The statements of C4: Personal opinion on reducing the SEC during software development

Code	Statement	Order
C4.9A	Addressing the energy consumption of our software should gain more attention.	16
C4.9B	The energy consumption of software is irrelevant.	23
C4.10A	I would like to reduce the energy consumption, if I am allowed to spend time on it.	27
C4.10B	Addressing the energy consumption would be a waste of time.	20

Table 5.8: The statements of C5: Social norm (e.g. SPO policy) on developing energy efficient software

Code	Statement	Order
C5.11A	The energy consumption of our software product is discussed during (in)formal meetings.	24
C5.11B	There is no discussion on the energy consumption of the software product in our team.	17
C5.12A	If other teams reduce the energy consumption of their software product, I would attempt it too.	21
C5.12B	Initiatives on energy reduction at other software products do not influence our actions on energy consumption.	28
C5.13A	Reducing the energy consumption would be a benefit to the SPO and to the customer.	18
C5.13B	I believe, nor the company or the customer would benefit from the energy reduction of software.	25

Table 5.9: The statements of C6: Other motives to reduce the SEC

Code	Statement	Order
C6.14A	Code optimization's to improve software qualities should be acknowledged and included in our backlog.	29
C6.14B	The backlog doesn't need to contain software qualities to improve the software.	22
C6.15A	The benefits of rewriting the code to reduce the energy exceed the costs.	26
C6.15B	The costs of rewriting the code, to reduce the energy consumption of our software, are too high compared to the benefits.	19

Table 5.10: Conversion to points

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
-2	-1	0	1	2

answering the same statement twice. After the feedback round, the order of the statements was adjusted. The original setup was to keep the order of the constructs and mix the statements within each construct. The idea was that the order of the constructs range from generic (energy consumption of software) to more concrete (possibilities to reduce the SEC). However, the limited number of questions didn't allow this construction and therefore the statements of the first three constructs and the last three constructs were mixed.

The third round was among four IT professionals of Centric with an average working experience of 3 years. The feedback mainly consisted of some unrelated questions regarding the electronic tool and not about the statements.

The statements are answered by 5-point Likert-items ranging from Strongly Disagree to Strongly Agree with the exception of one multiple choice statement. The Likert-items were converted into scores as shown in figure 5.10. With the conversion we were able to quantify the results and apply statistical analysis.

5.3.2 Measuring User-Acceptance

Besides the measurement of the awareness there was the measurement of the user-acceptance. The goal of this concept is to see whether the introduced dashboard aligns with the needs of the participants.

Theoretical Model

The user acceptance survey is inspired by the Unified Theory of Acceptance and Use of Technology (UTAUT) as proposed by Venkatesh, Morris, Davis, and Davis (2003). The UTAUT-model (5.7) is based on the elements of eight models about user acceptance and empirically validated.

The UTAUT-model (see fig 5.7) contains a set of constructs with a list of questions to measure the acceptance of a new technology. The constructs are measured by 29 questions which produce scores. These score can be compared in order to determine the acceptance of the new technology by the users.

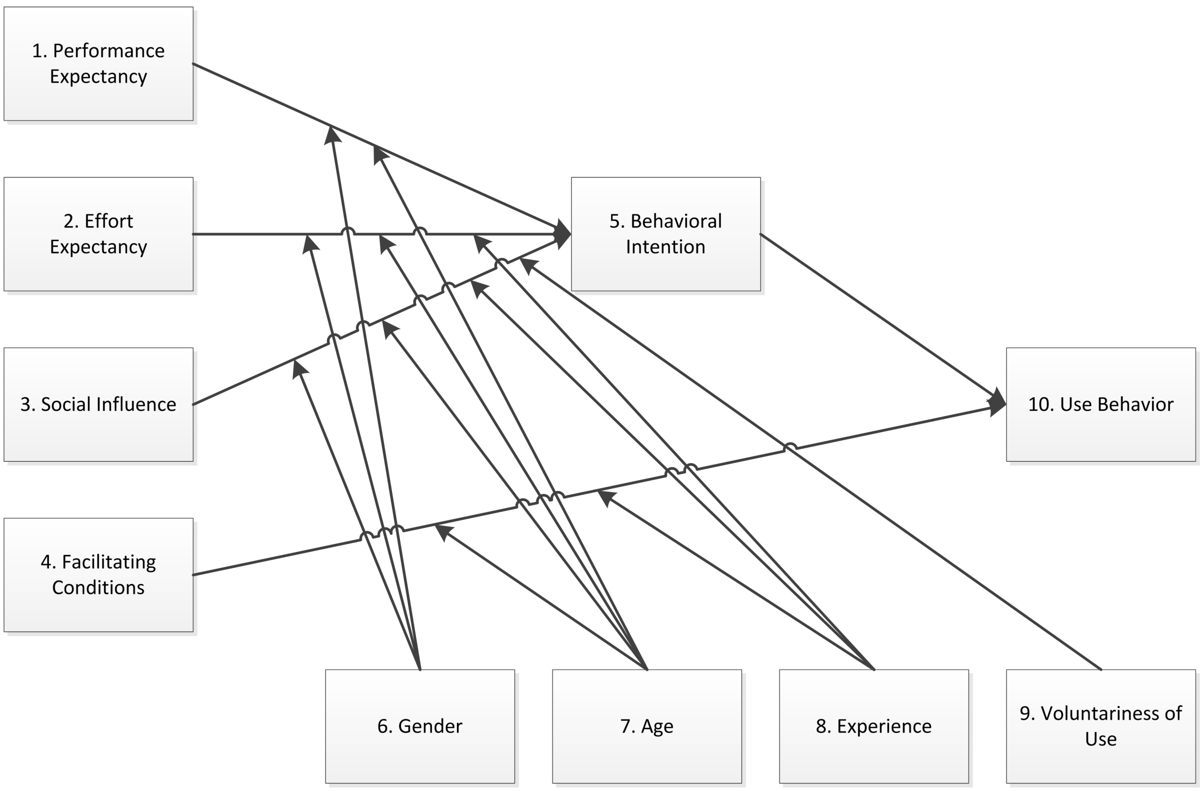


Figure 5.7: The UTAUT model derived from Venkatesh et al. (2003)

Figure 5.7 presents the UTAUT-model. The four constructs on the left and the construct floating in the middle determine the use behavior of the dashboard. The four constructs at the bottom are motivators which influence the other five constructs and should be taken into consideration.

Constructs

The empirical validation of the UTAUT-model filtered out the non-significant elements (constructs) from the original eight models. Even though the constructs are not significant, they still contain valuable elements about the acceptance of technology and therefore we created a new questionnaire inspired by the UTAUT-model.

Our questionnaire consists of the constructs from the UTAUT-model with the exception of *facilitating conditions*. The statements, to measure the acceptance of the energy dashboard, would be inappropriate. For example, item PBC2 'I have the resources necessary to use the system' of the construct *Facilitating Conditions*, would result in positive answers as the dashboard is hosted on an internal webserver free to access for every partic-

Table 5.11: Applied items of UTAUT

Construct	Items	Table
Performance Expectancy	U6, Ux	5.12
Effort Expectancy	EOU3, EOU6	5.13
Attitude towards using technology	A1, AF1, AF2, Affect1	5.14
Social Influence	SN1, SN2, SF2, SF4	5.15
Behavioral Intention	BI1, BI2, BI3	5.16

ipant. In addition, the items of the construct *Attitude towards using Technology*, would be a valuable insights, as the target audience consists of developers and other employees with a technical background, and measure their response to an experimental new technology. To conclude, we created a survey based on the following constructs:

1. Performance Expectancy (PE)
2. Effort Expectancy (EE)
3. Attitude towards using technology (AT)
4. Social influence (SI)
5. Behavioral intention to use the system (BI)

Performance Expectancy is the gain from using the artifact as expected by the users in his or her job. The *Effort Expectancy* encompasses the degree of ease associated with the use of the dashboard. *Attitude towards using technology* describes the view of the users towards the usage of a dashboard. *Social Influence* captures the social aspects of the working environment of the users. Finally *Behavioral intention to use the system* describes whether users plan to use the system.

Statements

Each construct defined in the previous section contains a set of questions to measure the construct. The questions were put together and reviewed by their applicability and value. Item EU4 'learning to operate the system is easy for me' of Effort Expectancy was removed, as operating the system would be an easy to accomplish task. Item Ux of Performance Expectancy was added to link the energy dashboard to the daily operations of the developers in line with item U6. Table 5.11 presents an overview of the item labels as applied in the work of Venkatesh et al. (2003) with the construct in the first column, the items in the second column and a reference to the tables with the corresponding statements.

Table 5.12: The statements of Performance Expectancy

Code	Statement	Order
U6	I find the dashboard useful in my job.	1
Ux	If I use the dashboard, I will increase my programming skills.	6

Table 5.13: The statements of Effort Expectancy

Code	Statement	Order
UOU3	My interaction with the dashboard is clear and understandable.	2
EOU6	I would find the dashboard easy to use.	7

Table 5.14: The statements of Attitude towards using Technology

Code	Statement	Order
A1	Using the dashboard is a good idea.	3
AF1	The dashboard makes work more interesting.	8
AF2	Working with the dashboard is fun.	11
Affect1	I like working with the dashboard.	14

Table 5.15: The statements of Social Influence

Code	Statement	Order
SN1	People who influence my behavior think that I should use the dashboard.	4
SN2	People who are important to me think that I should use the dashboard.	9
SF2	The senior management of this business has been helpful in the use of the dashboard.	12
Sf4	In general, the organization has supported the use of the dashboard.	15

Table 5.16: The statements of Behavioral Intention

Code	Statement	Order
BI1	I intend to use the dashboard in the next sprint.	5
BI2	I predict I will use the dashboard in the next sprint.	10
BI3	I plan to use the dashboard in the next sprint.	13

5.3.3 Versions

In total we created 29 statements to measure awareness and 15 to measure the user acceptance. Combined would

To reduce the impact of the questionnaire on the stakeholders we created different versions for different moments. There are three different versions:

- Version A, Awareness measurement based on C1, C4, C5 and C6.
- Version B, Awareness measurement based on C2 and C3 constructs.
- Version C, User acceptance test.

Version A consists of the less variable constructs C1, C4, C5 and C6 and are measured at the start and end of each case study. C1 targets the general consciousness of the SEC which changes the moment it is addressed by the survey. Addressing this construct for each successive sprint review will result in constant high scores. C4, C5 and C6 target the motivation to change behavior from different perspective which doesn't change in a short amount of time. Therefore it is only measured before and after the introduction of the dashboard. Version B consists of C2 and C3, targeting the consciousness of the individual participant. The consciousness is constantly stimulated by the stimulus and therefore measured at every sprint review. Version C targets the user acceptance and consists of all the acceptance constructs (PE, EE, AT, SI, BI). Version C can only be applied after the presentation of the dashboard.

5.4 Case Study Protocol

The case study protocol describes the execution of the multiple embedded case study. The two cases, one at a small development team and one at a large development team followed the same case study protocol. However, at some point there are deviations due to practical limitations.

5.4.1 Selection of the case

The first step of our research protocol is the selection of the cases. The inclusion criteria were based on three aspects: process, technology and the development team.

- An agile development method is implemented including reviews after each development iteration.

- At least five releases are available, four of which will be developed during the case study.
- The ability to deploy the software in a production-like configuration.
- Automated load tests are available.
- Commitment to fill in multiple surveys.

5.4.2 Preparation

The preparation phase consisted of three main activities prior to the case study itself.

Initial meeting

The first activity was to visit the product manager in order to get top level commitment and explain the details of our research. This activity was vital to the success of our case study as not everyone was eager to participate. The initial meeting provided the opportunity to align the schedules to determine the tested releases and the dates of the sprint review meetings. Finally it also provided a starting point to get into contact with specialists to setup the test environment.

Test Environment

With commitment from the product manager we are able to contact the specialist to setup a test environment with dedicated servers to simulate a production environment. The software product is installed on the server(s) together with Joulemeter. This application is developed by Microsoft Research and is able to create a powermodel of the machine. Based on this power model it can calculate the energy consumption of a monitored process with a granularity of one measurement per second in watt. The requirements of Joulemeter are:

- The operating system must be Windows 7.
- .NET Framework 3.5
- Battery pack or Watt's Up Pro energy meter for calibration

The requirements of Joulemeter introduces a problem as the operating systems of the servers are the Windows server editions. To circumvent this requirement we install Joulemeter on a Windows 7 system and copy the installation folder to the servers of the test environment. A Watt's Up Pro energy meter is connected by a USB connector

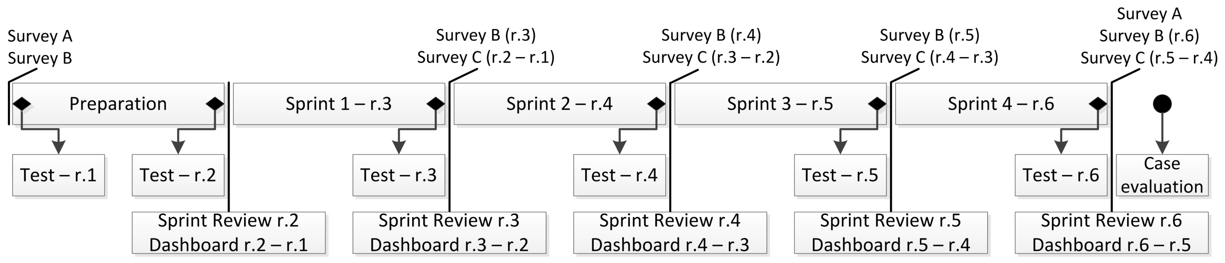


Figure 5.8: The case study planning including sprints, test periodss, sprint reviews, energy dashboard presentations and surveys.

to the server to re-calibrate Joulemeter. Finally, the folders were multiplied to create a Joulemeter instance for every monitored process on the same machine.

Test Scenario

We design the test scenarios in collaboration with the product manager and the test engineer of the software product. The test engineer provides the technical details in order to setup an automated test while the product manager approves on the tested functionality. The test scenario consists of the main functionality of the product, as testing every aspect of the product would be too extensive.

5.4.3 Execution

The schedule of the execution of the research protocol is shown in figure 5.8. The case starts with the execution of Survey A and B among the development team at the start of the preparation phase. After the survey starts the testing of release 1, which was delivered in the previous sprint. The development team delivers release 2 at the end of the preparation phase. Release 2 is tested in the test environment and the results are added to the dashboard together with the results of release 1. During the sprint review of release 2 we only present the dashboard to the participants. After the sprint review of release 2 starts sprint 1 with release 3. The release is tested at the end of the sprint and the results are compared with the results of the previous release on the dashboard. With the new dashboard we attend the sprint review meeting and ask the participants to fill in survey B and C. With survey B we look back at the dynamics regarding the SEC of the previous sprint and with survey C we measure the user acceptance of the presented dashboard from the previous sprint review. By starting with the survey instead of the dashboard, we force the participants to look back on the sprint without being biased by the dashboard. After collecting the surveys we present the new dashboard with the results of the last sprint. We explain the delta's we've found between the two software versions and ask the developers whether they are able to explain the results. The successive sprints of release four, five and six follow the same procedure with one exception to sprint review of release six. During the sprint review of release six we also ask the participants to fill in survey A.

Each software release was tested according to a standardized procedure consisting of six phases. First is the installation of the software version on the test server. Second is the execution of the test scenario and collecting all of the logs. Third is the ETL process of extracting, transforming and loading the right data and turn it into useful information. Fourth is the data analysis to calculate the metrics. Fifth is the creation of the dashboard. Sixth is the execution of the survey to capture the awareness.

Installation

The installation of the software product differs from product to product. The exact installations of both product are described in their own sections.

Execution of the test

The execution of the test was based on a declared protocol:

1. Reboot the server.
2. Start all instances of Joulemeter.
3. Start Performance Monitor.
4. Wait 20 minutes to let the server enter an idle state.
5. Execute the software product.
6. Execute the automated test.
7. Stop Performance Monitor and Joulemeter.
8. Collect all the logs.

ETL process

The ETL process consists of extracting, transforming and loading the data from the log files from all the Joulemeter instances and Performance Monitor. The log files of Joulemeter and Performance Monitor are .csv files. The first step is to extract all of the files from the servers to the workstation and create a copy of every file in order to save the original files. The second step is to transform the data from the Joulemeter .csv files by replacing every ',' into '.' and every ';' into ','. With the transformation Excel is able to spread the data into different columns. Third is to load all the data from the transformed files into one "workspace" .xlsx file. In the workspace file, all of the unnecessary data was removed for analysis.

Data analysis

The data analysis is applied in the Excel workspace file with custom formulas. The data analysis consists of calculating the defined metrics of the CPU, memory, hard disk, network, execution time and energy consumption.

Kansal et al. (2010) applied measurement by hardware utilization to determine the energy consumption of a virtual machine with the processors, memory banks, disk arrays, network cards and graphics cards as main components of a server, with the first three as the main consumers of energy. The lack of graphic cards in our test environments excludes this component as a metric. The other four are included to our set of metrics and expanded with *execution time* and *SEC*.

The CPU is represented by a CPU utilization score. It is based on the average CPU utilization of each monitored process during the execution of the task. The averages are summed to calculate the total utilization of the whole server. The result is divided by the number of logical cores of the processor. The formula for the CPU utilization score is presented in calculation 5.7 with i atleast 1 and max p for the total number of monitored processes, j for the record out of n records and k for the number of logical cores on the CPU of the machine.

$$CPU\,Utilization = \sum_{i=1}^{p} \frac{\frac{1}{n} \cdot \sum_{j=1}^{n} X_j}{k} \tag{5.7}$$

The memory is represented by memory utilization. The memory utilization is calculated by the sum of the average memory usage of each process on each server. The formula (i.e. 5.8) is similar to the formula of the CPU utilization with p for the total number of processes, j for the record out of n records.

$$Memory\,Utilization = \sum_{i=1}^{p} \frac{\sum_{j=1}^{n} X_j}{n} \tag{5.8}$$

The hard disk is represented by two metrics. The first metric is the average hard disk utilization and is calculated by subtracting the idle percentage of the full 100% over time. The result is the average hard disk utilization for each hard disk. Each server contains one hard disk and thus the result represents the hard disk utilization of the whole server.

$$HddUtilization = 100 - \frac{\sum_{j=1}^{n} X_j}{n} \tag{5.9}$$

The next metric is the average hard disk transfer rate of the number of bytes sent to (write) and retrieved (read) from the hard disk over time. The formula is provided in calculation 5.10. In this formula j represents the transfer rate record out of n records.

$$Hddtransfer = \frac{\sum_{j=1}^{n} Xj}{n} \tag{5.10}$$

Network utilization represents the numbers of bits per second transferred through the network port(s). The calculation (see 5.11) is based on the sum of all the records j divided by n number of records for each network port np.

$$NetworkUtilization = \sum_{i=1}^{np} \frac{\sum_{j=1}^{n} X_j}{n} \tag{5.11}$$

Execution Time represents the average time for the execution of completing the task. The formula for the calculation is presented in 5.12 with each run i having b as the end timestamp, a as the start timestamp and n as the number of runs.

$$ExecutionTime = \frac{\sum_{i=1}^{n} X_b - X_a}{n} \tag{5.12}$$

Energy consumption is presented in joule (J) but Joulemeter measures the power in watt (W) also known as joule/second. The calculation is to sum all of measurements i for n measurements as provided in 5.13.

$$Energyconsumption = \sum_{i=1}^{n} X_i \tag{5.13}$$

Dashboard

The data analysis provides the required values for the energy dashboard. The values are hard coded saved into the HTML file. Based on the input, a custom developed JavaScript function draws the radar chart on an HTML canvas element.

Survey

The execution of the survey consists of three parts. First is the survey filled in by the participants. Second is the presentation of the dashboard with the results from the last sprint. Third is the semi-structured discussion about the results.

Data Analysis Procedure

The data analysis procedure encompasses the statistical analysis of the gathered survey. A quantitative approach would have been preferred, however the limited number of participants (N=5) would question the statistical significance. Therefore, we follow a qualitative approach to describe the developments during the case studies. The acquired data from the survey is of the ordinal scale.

The data analysis of the survey data consists of three steps. First, the data was extracted from the original file into an excel workbook. Second, The numbers were converted from 1 (Strongly Disagree) to 5 (Strongly Agree), as the survey tool reported the output from 1 to 5, into -2 to +2. Third, the encoded data was loaded into a matrix for further analysis. The transformation of the data from (dis)agreement to negative and positive numbers keeps the original nature of the answer, as a disagreement is represented by a negative number. This approach provides an improved interpretation of the opinions of the participants. At the same time it may reduce the readability due to the representation of zero in graphs.

Each statement of the survey consists of a negatively and positively formulated item. A Spearman's rank-order correlation was run to determine the relationship between the positive and negative items on the data of survey A from both cases. Preliminary analysis showed the relationship to be monotonic, as assessed by visual inspection of a scatterplot. There was a medium to strong negative correlation between the positive and negative answers, $r_s(180)$= -.526, $p < .001$ for DocGen and $r_s(308)$= -.437, $p < .001$ for RetailSystem.

The score for awareness is calculated by the sum of all constructs, of which each construct consists of the sum of all its item scores among all participants.

The analysis of the data obtained from the RetailSystem case was based on the data from 12 participants. The digital survey had a lower response rate compared to the DocGen case. Therefore we created the inclusion criteria to compare the same group of participants among the sprint reviews. The criteria were based on the completion of the survey at SR-r.1 and SR-r.6 and the completion of atleast 3 out of 4 survey in between.

5.4.4 Ethics

The case study involves participation of humans and therefore certain ethical aspects need to be taken into consideration. The first one is the consensus with the participants and the second is the privacy and confidentiality regarding the data.

Consensus with the participants is hard to realize due to the biased responses with the survey. If all of the participants were fully informed about the goal of the survey, the validity of the research would be jeopardized. To avoid this situation we informed the product manager of the development team(s) to discuss the participation.

The privacy and confidentiality of the participants were assured by anonymizing the answers of the participants. All participants received a number at the start of the case study. This number was written down at the top of every questionnaire. Only one person had access to the complete list with numbers and names and agreed not to share this data with anyone else. The confidentiality in this context is debatable, as one participant mentioned, as answers could be traced back to a respondent. However, it was the only option to keep track of the development of every participant and to keep the response rate at a high level.

Chapter 6

Case DocGen

The first case of the embedded case study involves a document generator called "DocGen". The next sections start with an introduction with the details of the software product and the development team. After the introduction follows the results section with all of the obtained results from the awareness and acceptance surveys.

6.1 Introduction

DocGen is an on-premise application hosted at a customer on-site or in a central datacenter. Its main functionality is to generate large quantities of standard documents with custom fields and to archive or print the result. DocGen is used by more than 900 end-users divided over 300 (mostly governmental) organizations in The Netherlands and generating over 30 million documents on yearly base.

6.1.1 Architecture

The application has a two-tier architecture with an application server and a database server. Figure 6.1 presents the computational stack of the application. The hardware layer consists of the hardware of table 6.1. On top of both hardware stacks runs Windows Server 2008 R2 Standard Edition 64-bit SP1. The DocGen application server runs on top of the operating system and consists of roughly five million lines of code written in C++. On top of the DocGen application server runs the instance used by the user. The DocGen database server hosts an Oracle Server. The Oracle Server has an Oracle Database which contains a DocGen Schema, containing the schema of the application. The Data schema contains of the tables with the data from the user.

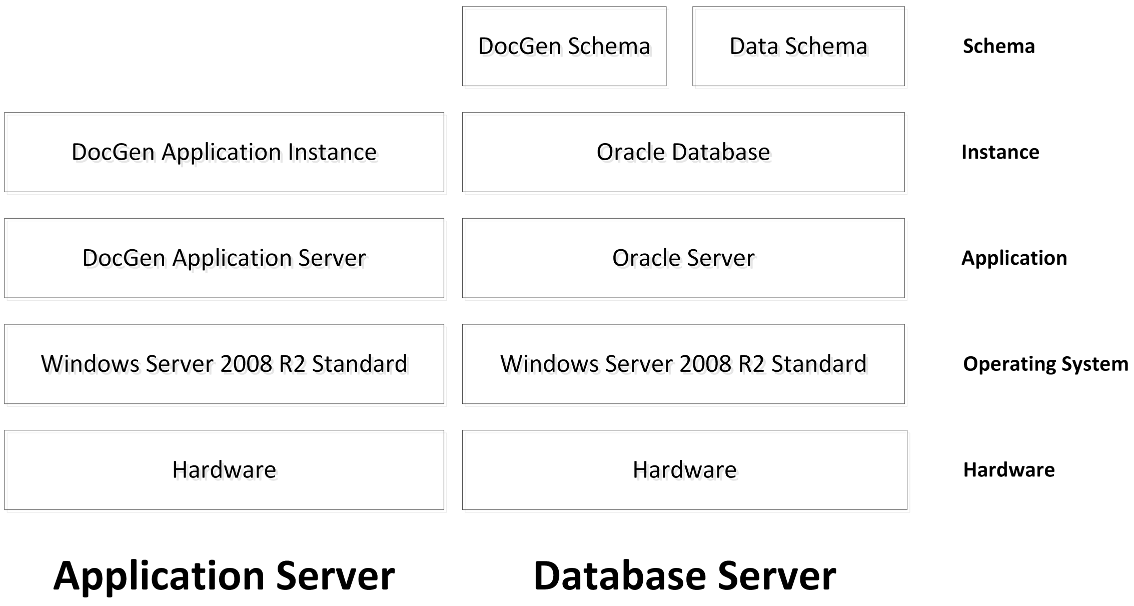


Figure 6.1: Computational stack of DocGen

In cooperation with the software architect we identified five processes to be monitored during run-time. On the database server runs the process *oracle.exe* as the database of the application. On the application server run *Interface.exe*, *Connector.exe*, *Server.exe* and *Run.exe*.

6.1.2 The Development Team

The development team consists of four developers, one tester and one software architect lead by one product manager in a scrum team located at one of the main offices. The

Table 6.1: Hardware configuration DocGen servers

Property	Application Server	Database Server
Brand/model	HP DL380	
Processor	2x Intel Xeon E5335 @ 2.0 GHz	
Number Proc	2	1
FDB / TDP	1333 Mhz / 80 w	
Chipset	Intel 5000P	
Memory	8192 MBytes FB-DDR2	
OS	Windows Server 2008 R2 Standard 64-bit SP1	

software architect and the product manager were excluded from the surveys as they were fully informed about the case study and therefore biased.

A development sprint lasts for three weeks and a new product version is released after every two sprints. Besides the sprint releases are the daily builds.

6.1.3 Preparation

The preparation of the case study started with informing the involved parties to get top-level support. After this activity we started the initial meeting with the stakeholders.

Initial meeting

The initial meeting was with the product manager and the test engineer of DocGen. During the meeting we discussed the goals and the vision of measuring the energy consumption of software. The product manager gave its permission to conduct the research and directed the rest of the communication to the test engineer who also had the role of scrum master in the development team.

The Test Environment

The development team doesn't have a dedicated test environment and tests locally on the development machines. To simulate a customer environment we installed the application and database on two separated machines in a hosting environment. Joulemeter was installed on both machines and calibrated with a Watts Up Pro energy meter. A third machine executed Performance Monitor and monitored both servers remotely via the network.

Test Scenario

The test scenario was created in collaboration with the test engineer. DocGen is mainly used to generate standard documents with custom fields, for instance name and address. The scenario we created was to generate a document with 17 custom fields for 258 different addressee. The test consisted of generating and deleting this job 40 times. The test was automated with a script to generate the documents and delete them afterwards. The total execution of the test required seven hours.

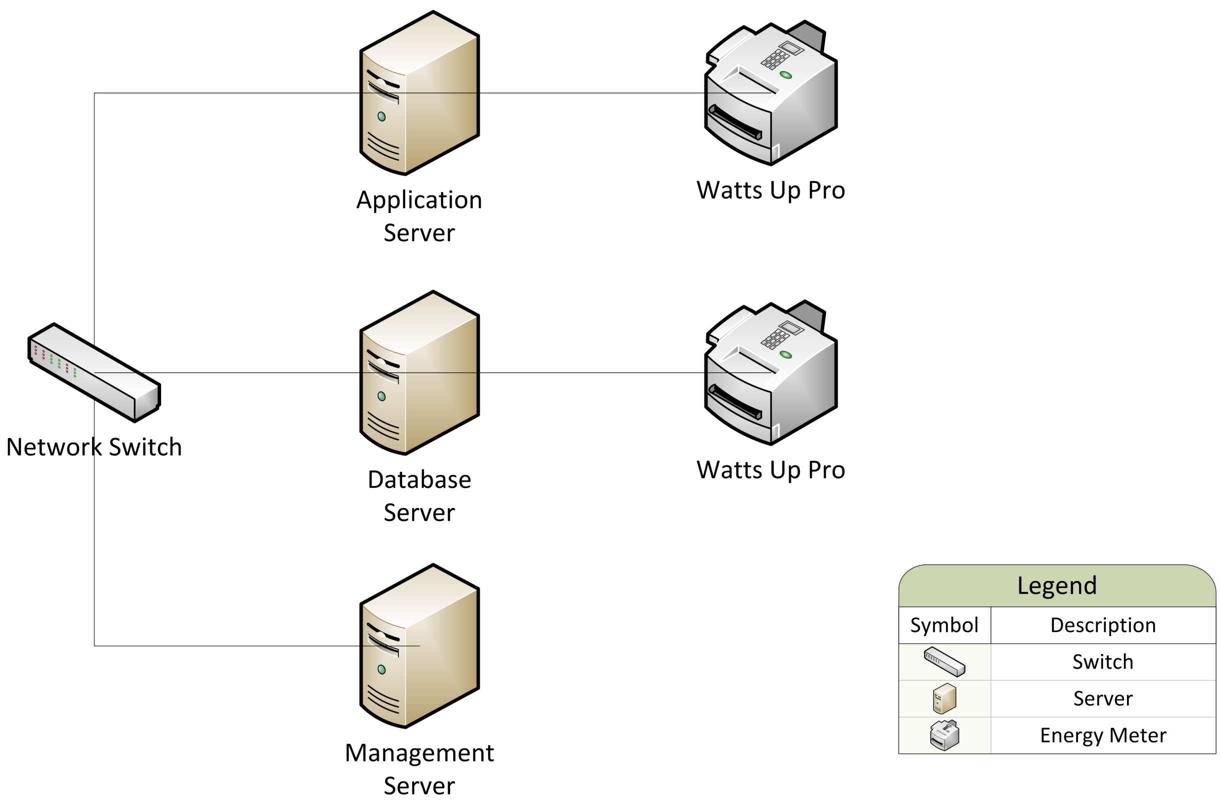


Figure 6.2: Server setup DocGen

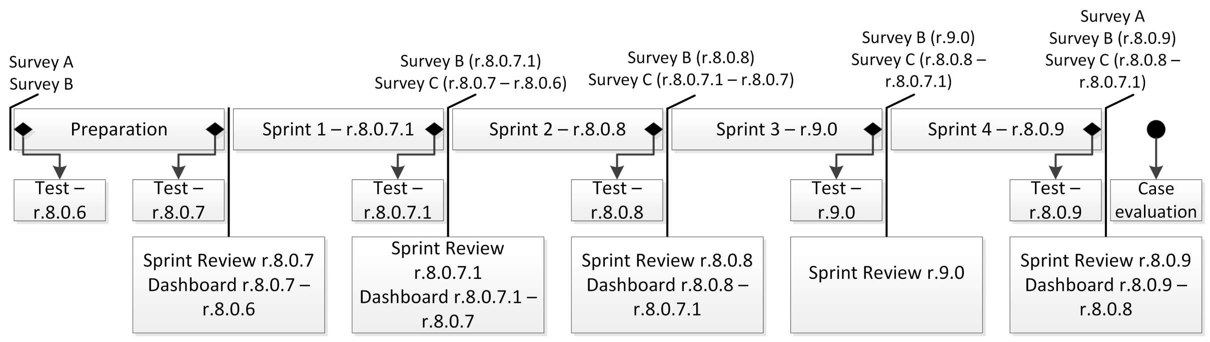

Survey A
Survey B
Preparation
Test – r.8.0.6
Test – r.8.0.7
Sprint Review r.8.0.7 Dashboard r.8.0.7 – r.8.0.6
Survey B (r.8.0.7.1)
Survey C (r.8.0.7 – r.8.0.6)
Sprint 1 – r.8.0.7.1
Test – r.8.0.7.1
Sprint Review r.8.0.7.1 Dashboard r.8.0.7.1 – r.8.0.7
Survey B (r.8.0.8)
Survey C (r.8.0.7.1 – r.8.0.7)
Sprint 2 – r.8.0.8
Test – r.8.0.8
Sprint Review r.8.0.8 Dashboard r.8.0.8 – r.8.0.7.1
Survey B (r.9.0)
Survey C (r.8.0.8 – r.8.0.7.1)
Sprint 3 – r.9.0
Test – r.9.0
Sprint Review r.9.0
Survey A
Survey B (r.8.0.9)
Survey C (r.8.0.8 – r.8.0.7.1)
Sprint 4 – r.8.0.9
Test – r.8.0.9
Sprint Review r.8.0.9 Dashboard r.8.0.9 – r.8.0.8
Case evaluation

Figure 6.3: Case study execution DocGen

6.1.4 Execution

The execution of the DocGen case study followed a slightly different compared to the general schedule. Figure 6.1.4 provides the schedule of the execution of the case study DocGen. In the case of DocGen there was no survey during the first sprint review. Based on our insights we were able to introduce this extra survey at the second case. The DocGen case also misses one dashboard with an extra survey B and C. During the case study the development team delivered release 9.0 as a new release. However, after excessive bug fixing during the installation, the release was still unstable and not suitable for our test. Therefore we were unable to present the dashboard and canceled the meeting.

The rest of the execution was conducted according to the six phases of the case study protocol with the installation, testing, ETL process, data analysis, dashboard generation and survey execution.

Installation

The installation of the software only occurred on the application server. The database server only received cumulative updates by the system administrators. The versions were installed with the use of an installer (.exe) and saved in separated directories. Each version created a new Windows service to start and stop the application. This approach introduced the possibility to install different versions on the same machine.

Execution of the test

The execution of the test was automated with a test. The first instruction of the test was to login to the application. This step started the monitored processes and provided the interface with the progress of the jobs. The next step was the execution of the test script:

1. Login with user credentials in DocGen.
2. Wait for 10 seconds.
3. Delete all previous 258 documents.
4. Wait for 60 seconds.
5. Generate 258 new documents.
6. Wait for 420 seconds.

The script executed the instructions row by row and not when an instruction was finished. Therefore it only paused when it was forced to do. Therefore the deletion of the document occurred during the sixty seconds of the pause and the generation of the documents during the 420 seconds of the pause. In practice the deletion lasted for approximately twenty five seconds and the generation around 360 seconds. The resulting time allowed to server to go to an idle state before the next round started.

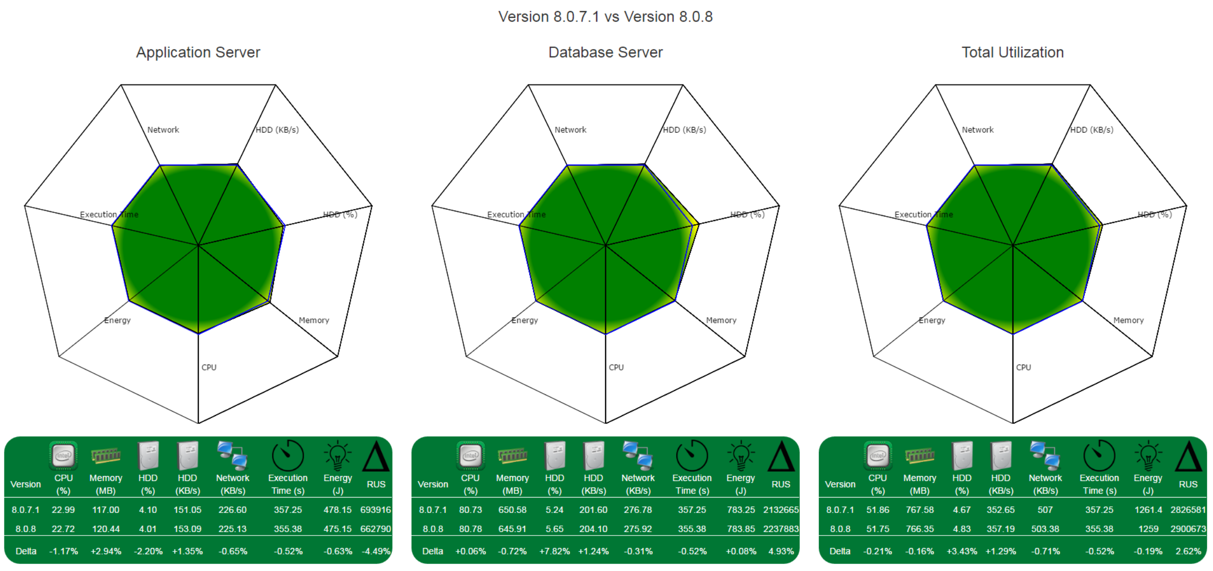
The login of the test script occured once. The rest of the steps were forty times executed. The manual login and the script login results in a double login, but not a double amount of running processes. The manual login was required in order to monitor the interface. The scripted login does not trigger the launch of the interface which would be left out of the test. However, the interface is an important aspect of a representative test case and therefore we included it in the test.

ETL

The log files were copied to the workstation and converted in order to become readable .csv files for Excel. The data from the files was loaded in the workspace file and reduced to the minimum required data. This step reduced the size of the data by 80%.

Data analysis

The applied metrics were calculate for the application server, the database server and for DocGen in general. With this approach we provided the resource utilization and energy consumption of both machines and of the overall application.



Application Server

Version	CPU (%)	Memory (MB)	HDD (%)	HDD (KB/s)	Network (KB/s)	Execution Time (s)	Energy (J)	RUS
8.0.7.1	22.99	117.00	4.10	151.05	226.60	357.25	478.15	693916
8.0.8	22.72	120.44	4.01	153.09	225.13	355.38	475.15	662790
Delta	-1.17%	+2.94%	-2.20%	+1.35%	-0.65%	-0.52%	-0.63%	-4.49%

Database Server

Version	CPU (%)	Memory (MB)	HDD (%)	HDD (KB/s)	Network (KB/s)	Execution Time (s)	Energy (J)	RUS
8.0.7.1	80.73	650.58	5.24	201.60	276.78	357.25	783.25	2132665
8.0.8	80.78	645.91	5.65	204.10	275.92	355.38	783.85	2237883
Delta	+0.06%	-0.72%	+7.82%	+1.24%	-0.31%	-0.52%	+0.08%	4.93%

Total Utilization

Version	CPU (%)	Memory (MB)	HDD (%)	HDD (KB/s)	Network (KB/s)	Execution Time (s)	Energy (J)	RUS
8.0.7.1	51.86	767.58	4.67	352.65	507	357.25	1261.4	2826581
8.0.8	51.75	766.35	4.83	357.19	503.38	355.38	1259	2900673
Delta	-0.21%	-0.16%	+3.43%	+1.29%	-0.71%	-0.52%	-0.19%	2.62%

Figure 6.4: Dashboard of DocGen for version 8.0.8 to version 8.0.7.1

Dashboard

The dashboard of DocGen consists of three graphs and three tables. The first graph represents the application server, the second graph represents the database server and the third graph represents the total utilization over both machines by DocGen. Below each graph is a table with the data of each graph. An example is visible in figure 6.4.

The tables in figure 6.4 also present other data than the data from the graphs. The last column of every table presents the Resource Utilization Score of the respective version. The delta of the RUS indicates the general changes between the two versions of the software product.

6.1.5 Survey

The collection of the awareness on energy consumption was conducted by means of the survey printed on paper. Each participant wrote its number on top of the paper en filled in the questionnaire. After gathering the papers it was time for the presentation of the dashboard. Filling in the questionnaire before introduction of the dashboard forced the participants to look back on the dynamics of the last sprint without being influenced by the results from the dashboard. The dashboard was presented with a beamer. At the same time the presenter explained the differences in performance and energy metrics between the two product versions.

After the presentation of the dashboard were the discussions. The presenter asked the participants for their opinion of their results. The discussions differed from presentation to presentation. At the start of the case study, the participants were curious to the results and discussed suggestions to improve their own software product. The discussion of the second presentation focused on the validity of the measurements. The results of the measurements did not align with the expectations and results from the last sprint and therefore the participants were skeptic. The third presentation was again about the validity of the measurements but this time the results were aligned with the expectations. The differences between the two software versions (8.0.7.1 vs 8.0.8) were only minimal. At a later moment the scrum master explained the skepticism of the team. The dashboard caused friction in the team as the developers were judged while they felt unable to address the energy and performance aspects due to the newly implemented functionality. Finally, the fourth presentation provided more agreement as the results could be related to the implemented functionality.

6.2 Results DocGen

This chapter describes the results of the survey. The first section starts with the results from before and after the introduction of the energy dashboard. The second section describes the development of C2 and C3 over the sprint reviews. The third section describes the user acceptance scores.

6.2.1 Awareness before & after

The total awareness score decreased from +23 at SR-r.1 to +3 at SR-r.6. Figure 6.2.1 presents the scores per construct of survey A and B at the start (SR-r.1) and end (SR-r.6) of the case study.

The scores on C1 (*Consciousness of the energy consumption of software* increases from -2 to +5. C2 (*Consciousness of the relevance of creating energy efficient software* decreases from +6 at SR-r.1 to -1 at SR-r.6. The scores on C3 (*Consciousness of one's possibilities to reduce the SEC*, decreased from +4 at SR-r.1 to +1 at SR-r.6. C4 (*Personal opinion on reducing the SEC during software development* starts with +9 but ends with a score of +1 at SR-r.6. C5 (*Social norm on developing energy efficient software*), starts with a score of 0 at SR-r.1 and ends at SR-r.6 with +1. C6 (*Other motives to reduce the SEC* shows a decline in score from +6 to -4.

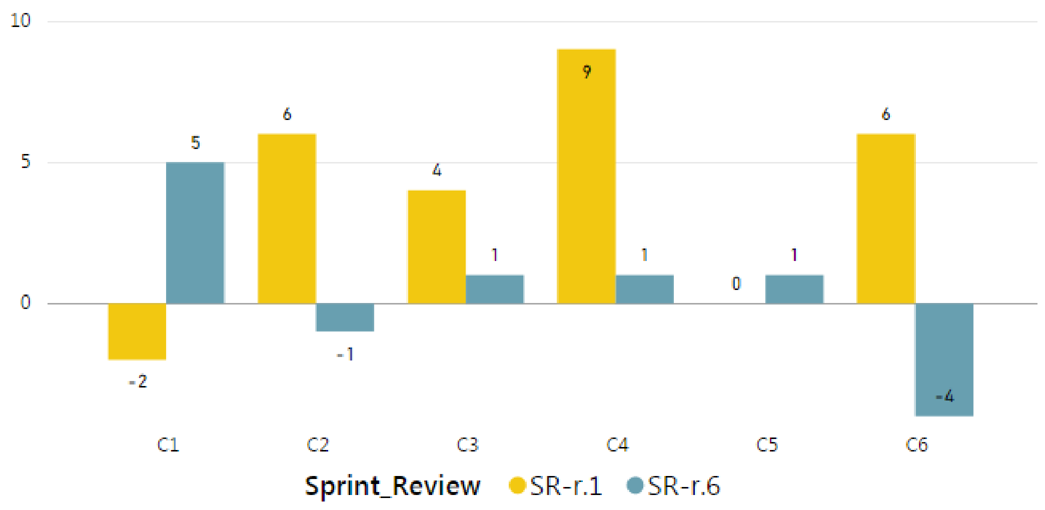


Figure 6.5: DocGen awareness scores per construct over the sprint reviews

C1 - Consciousness of the energy consumption of software

Figure 6.6 presents the development of scores among the questions of C1 at the start and end of the case study. Both questions show a change in scores between the two points in time. The question *Before this project, I wondered multiple times about the energy consumption of our software development environment* changes from -3 to +2. The second question, *I want to determine the energy consumption of our software development environment*, changes from a +1 to +3.

C2 - Consciousness of the relevance of creating energy efficient software

Figure 6.7 presents the change of scores among the questions of C2 at the start and end of the case study. The first question, *I expect that software has a large influence on the energy usage*, starts with a score of +1 at SR-r.1 and ends with a score of 0 at SR-r.6. The second question of C2, *I would like to know the energy consumption of our software product*, is on the right side of the figure. The initial score at SR-r.1 starts at +5 but decreases at SR-r.6 with -1.

C3 - Consciousness of one's possibilities to reduce the SEC

Figure 6.8 presents the scores of the questions of C3 among the sprint reviews. The first statement on the left (*If I had more time to work on the code, I would be able to*

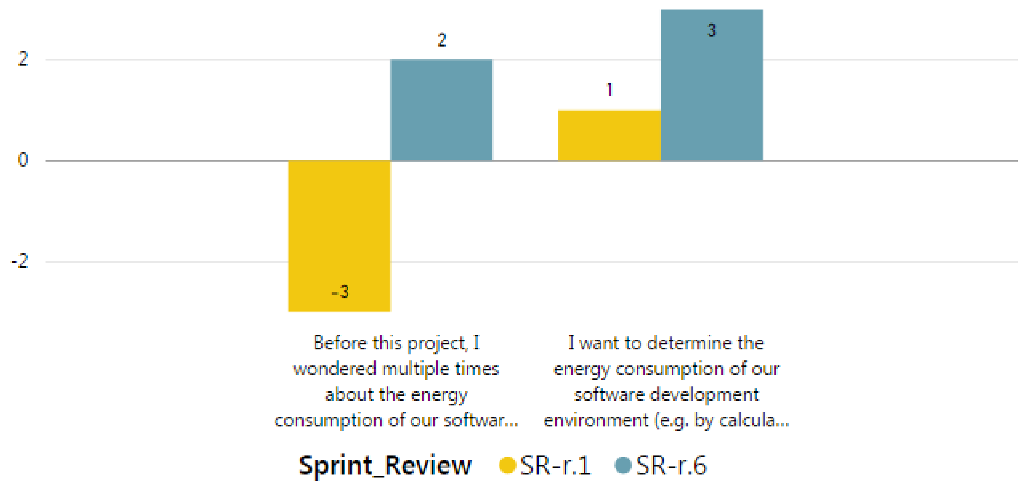


Figure 6.6: DocGen awareness scores of C1 per statement at SR-r.1 and SR-r.6

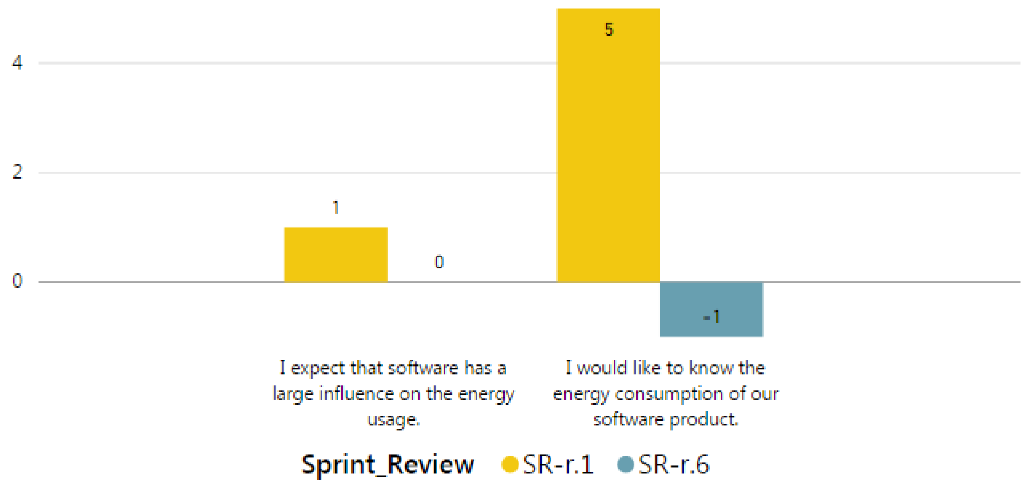


Figure 6.7: DocGen awareness scores of C2 per statement at SR-r.1 and SR-r.6

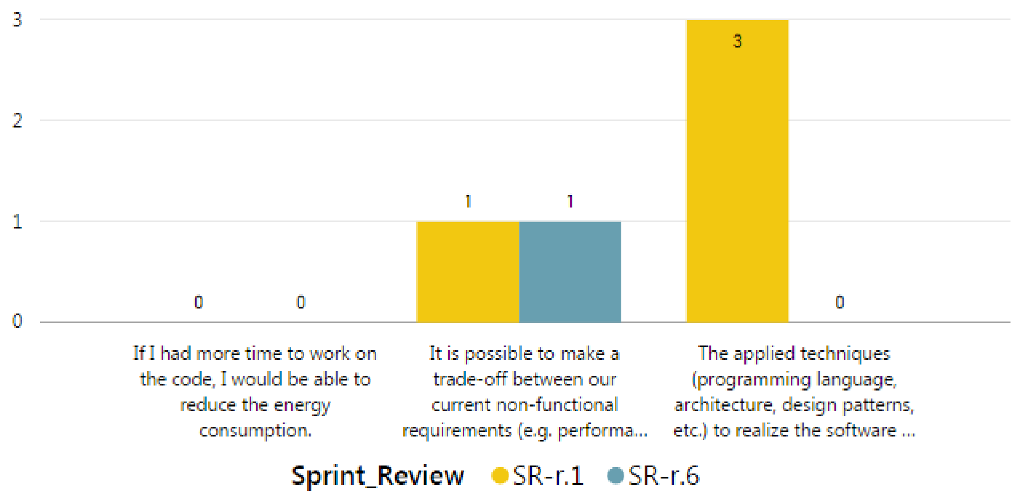


Figure 6.8: DocGen awareness scores of C3 per statement at SR-r.1 and SR-r.6

reduce the energy consumption), starts and ends with a score of 0. The second statement (*It is possible to make a trade-off between our current non-functional requirements (e.g. performance) and the energy consumption of our software)*, starts with a +1 and ends with a +1. The third statement (*The applied techniques (programming language, architecture, design patterns, etc.) to realize the software product, allow for the reduction of energy consumption*, starts with a score of +3 at the start of the case study but decreases to a score of 0.

Besides the statements, C3 also contained one specific statement with multiple choice answers: *The energy consumption should be addressed by: A)Developers B)Software Architects C)Developers and Software Architects D)Other*. Four of the five participants answered C and one participants answered C + D with the addition of *Testers* and *Product Owners*. The answers of SR-r.6 were identical to SR-r.1.

C4 - Personal opinion on reducing the SEC during software development

Figure 6.9 presents the scores per statement on C4. The first statement (*Addressing the energy consumption of our software should gain more attention*) starts with +3 at SR-r.1 and has become 0 at SR-r.6. The other statement (*I would like to reduce the energy consumptio, if I am allowed to spend time on it*) starts at +6 and ends at +1.

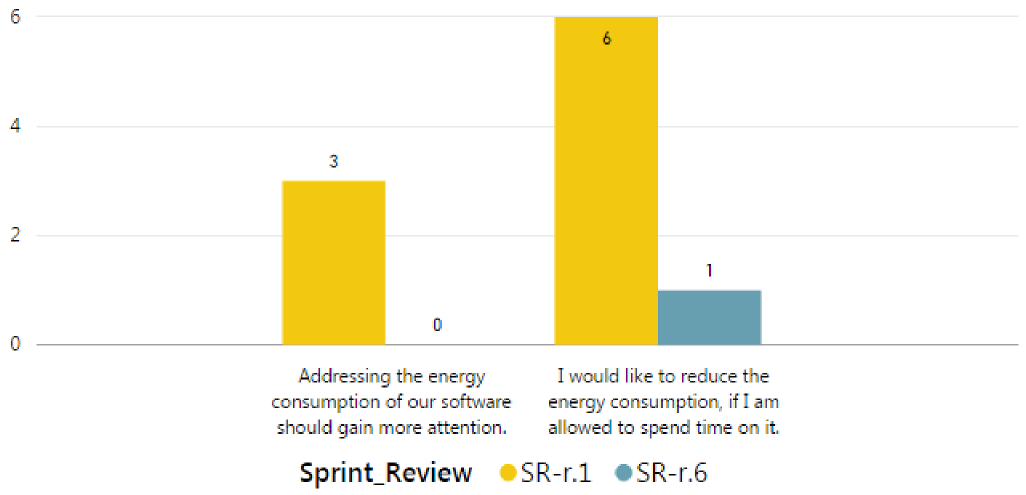


Figure 6.9: DocGen awareness scores of C4 per statement at SR-r.1 and SR-r.6

C5 - Social Influence

Figure 6.10 presents the scores per statement on C5. The first statement, *if other teams reduce the energy consumption of their software, I would attempt it too*, increases from -2 to +2 at the end of the case study. The second statement, *Reducing the energy consumption would be a benefit to Centric and the customer*, decreases from +5 to +3. The third statement, *The energy consumption of our software product is discussed during (in)formal meetings*, starts with -3 and becomes -4 at the end of the case study.

C6 - Other motives to reduce the SEC

Figure 6.11 presents the scores per statement on C6. The first statement, *Code optimizations to improve non-functional requirements should be acknowledged and included in our backlog*, starts with a score of +6 at SR-r.1 and has become -3 at SR-r.6. The other statement of C6 (*The benefits of rewriting the code to reduce the energy exceed the costs*) starts with 0 at SR-r.1 and has become -1 at SR-r.6.

6.2.2 Development over time

Survey B measured the constructs C2 and C3 at all of the sprint reviews. An overview of the scores on C2 and C3 at all four sprint reviews is presented in figure 6.12. The graphs show a change in scores among the two constructs. C2 starts with +6 at SR-r.1,

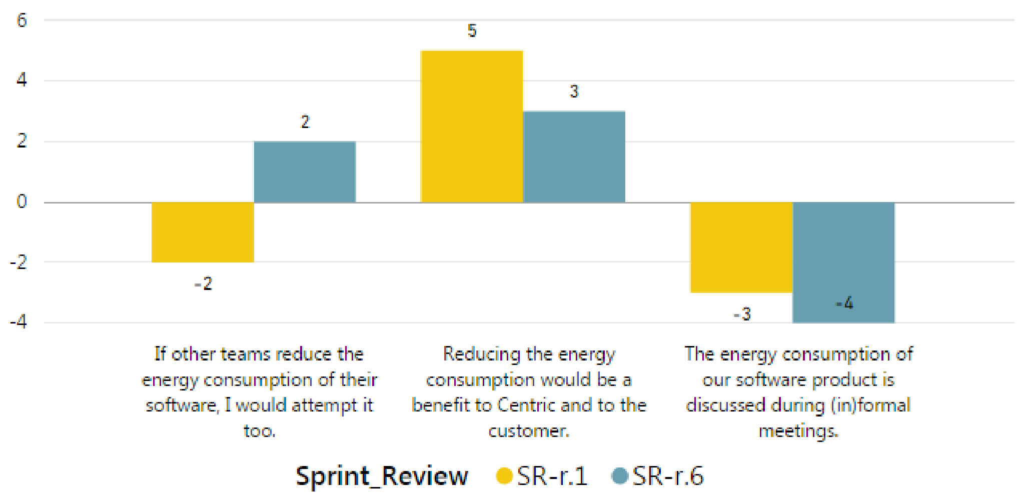


Figure 6.10: DocGen awareness scores of C5 per statement at SR-r.1 and SR-r.6

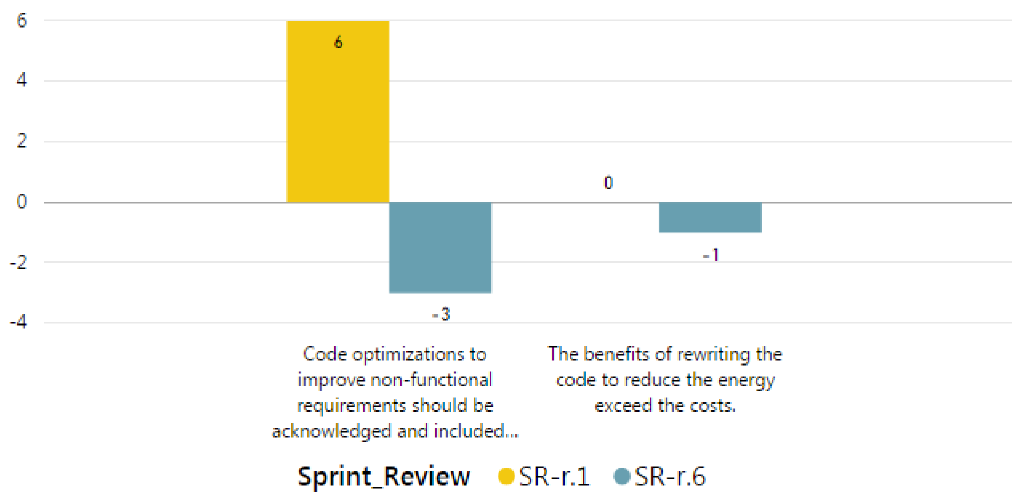


Figure 6.11: DocGen awareness scores of C6 per statement at SR-r.1 and SR-r.6

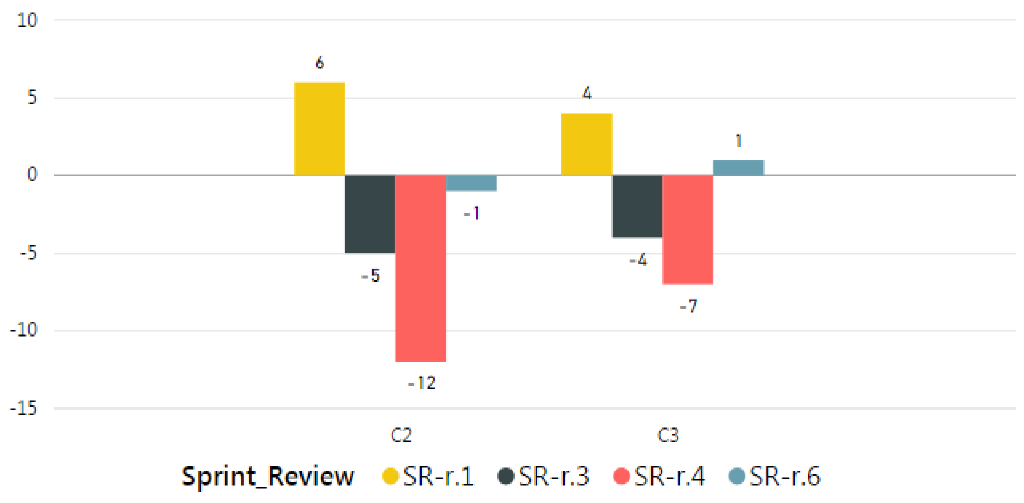


Figure 6.12: DocGen awareness scores on C2 and C3 among all sprint reviews

decreases to -5 at SR-r.3 and decreases even more to -12 at SR-r.4, to increase again to -1 at SR-r.6. A similar pattern occurs at C3 with a starting score of +4, followed by two decreases at SR-r.3 (-6) and SR-r.4 (-7), to increase at the end of the study (+1).

The scores per statement of survey B is presented in fig 6.13. The first statement (*I expect that software has a large influence on the energy usage*), changes from +1 to -1, -6 and 0 in the successive sprint reviews. The second statement (*I would like to know the energy consumption of our software product*) changes from +5 to -4, -6 and -1. The third statement (*If I had more time to work on the code I would able to reduce the energy consumption*) changes from 0 to -2, -2 and 0 again. The fourth statement (*It is possible to make a trade-off between our current non-functional requirements and the energy consumption*), changes from +1 to -1, -2 and +1 again. Finally the fifth statement (*The applied techniques () allow for the reduction of the energy consumption*), changes from +3 to -1, -3 and 0.

6.2.3 User Acceptance

The measurement of user acceptance was collected with survey C during the sprint review meetings. There were three measurements at sprint review SR-r.3, SR-r.4 and SR-r.6. Note that the user acceptance of P1 and P4 is missing at SR-r.4.

Figure 6.2.3 presents the acceptance scores per construct. A first glance at the graphs shows a decreasing score over time among most of the constructs.

AT, *Attitude towards using Technology*, starts with +6 at SR-r.3, decreases to +5 at

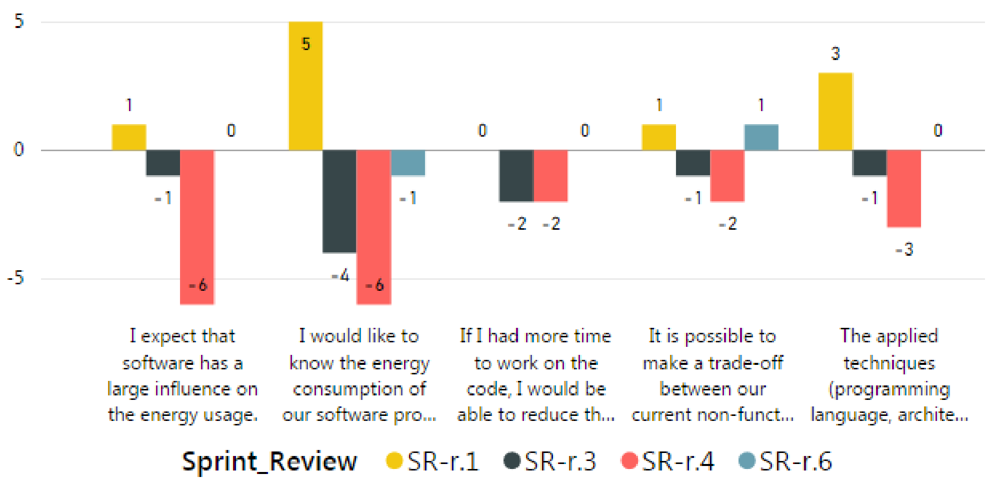


Figure 6.13: DocGen awareness scores of C2 and C3 per question at all sprint reviews

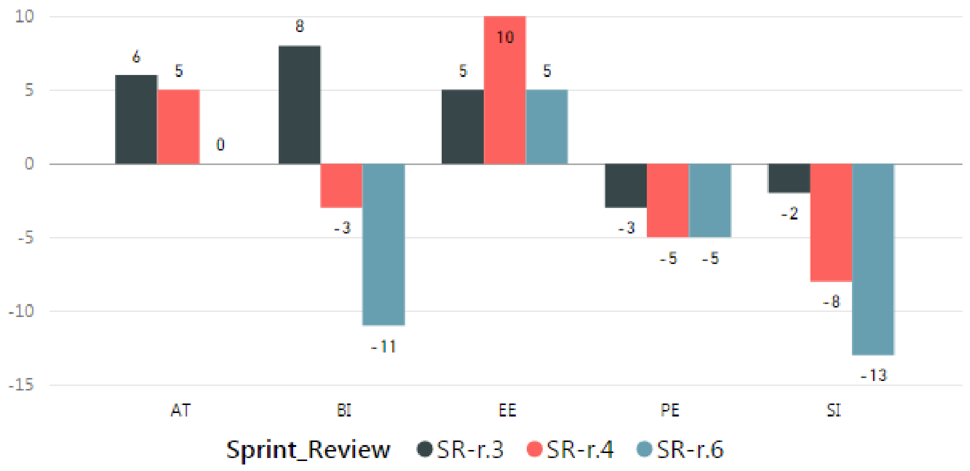


Figure 6.14: DocGen acceptance scores of all constructs

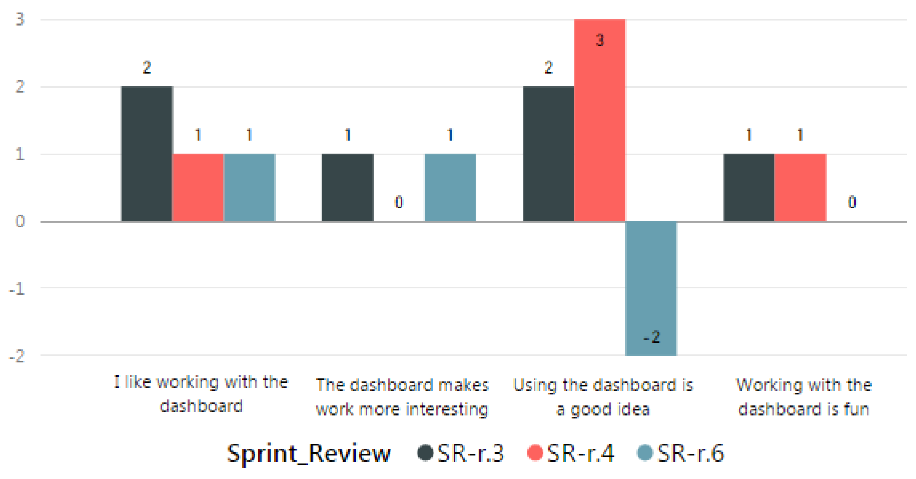


Figure 6.15: DocGen acceptance scores on Attitude toward using technology

SR-r.4 and becomes 0 at SR-r.6. BI, *Behavioral Intention*, starts with +8, decreases to -3 at SR-r.4 and -11 at SR-r.6. EE, *Effort Expectancy*, starts with +5, increases at SR-r.4 to +10, but decreases to +5 at SR-r.6. PE, *Performance Expectancy*, starts with -3, decreases at SR-r.4 to -5 and remains at the same score at SR-r.6. SI, *Social Influence*, starts at -2 but decreases every successive sprint to end at -13.

Attitude toward using technology

Figure 6.15 presents the scores on *Attitude towards using technology* (AT). The first question (*I like working with the dashboard*, starts with +2 which decreases at SR-r.4 to +1 and remains stable at SR-r.6 with +1. The second question (*The dashboard makes work more interesting*, starts with +1, becomes 0 at SR-r.4 and has become +1 again at SR-r.6. The third question (*Using the dashboard is a good idea*, shows large changes in scores. At SR-r.3 it is +2, which increases at SR-r.4 to +3 but decreased to -2 at SR-r.6 . The fourth question (*Working with the dashboard is fun*), starts with +1 at SR-r.3 and SR-r.4 and decreases to 0 at SR-r.6.

Behavioral Intention

Figure 6.16 presents the scores on the questions of behavioral intention. The first statement (*I intend to use the dashboard in the next sprint*), starts with +2, decreases to -1 at SR-r.4 and -3 at SR-r.6. The second statement (*I plan to use the dashboard in the next sprint*), starts at +3, decreases to -1 at SR-r.4 and ends with -3. The third statement (*I*

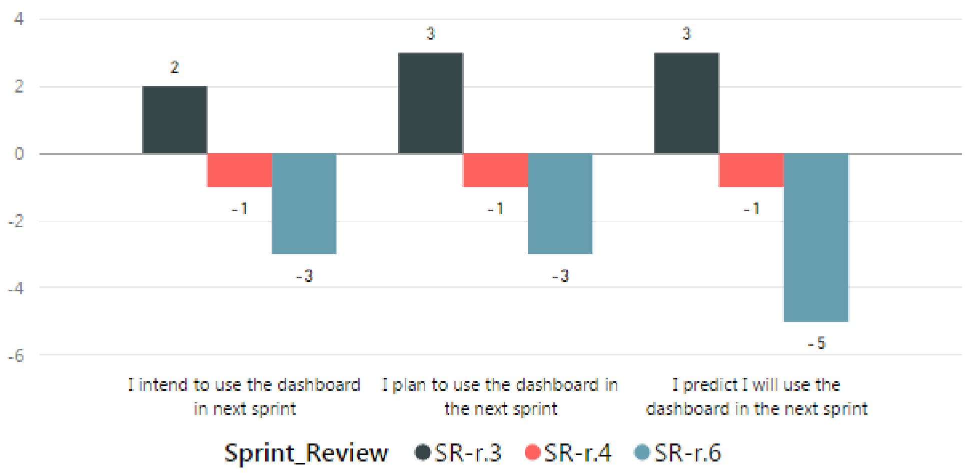


Figure 6.16: DocGen acceptance scores on Behavioral Intention

predict I will use the dashboard in the next sprint), starts at +3, decreases to -1 and ends with -5.

Effort Expectancy

Figure 6.17 presents the scores of each question from the construct *Effort Expectancy*. The graphs show a peculiar trend as the scores increase and decrease within the same question. This trend could be caused by the absence of two participants at SR-r.3. Another observation is the absence of negative scores. The first statement (*I find the dashboard easy to use.*), starts with +3, increases to +5 and decreases again to +3. The second statement (*My interaction with the dashboard is clear and understandable.*), starts with +2, increases to +5 and ends with +2.

Performance Expectancy

Figure 6.18 presents the scores on the questions of construct *Performance Expectancy*. The graphs in the figure show clear trends among the questions. The first question (*I find the dashboard useful in my job*) starts with 1 at SR-r.3 but decreases at the successive sprint reviews to -2 (SR-r.4) and -3 (SR-r.6). The second question (*If I use the dashboard, I will increase my programming skills*), shows the opposite trend as the score of SR-r.3 starts at -4 but increases to -3 (SR-r.4) and -2 (SR-r.6).

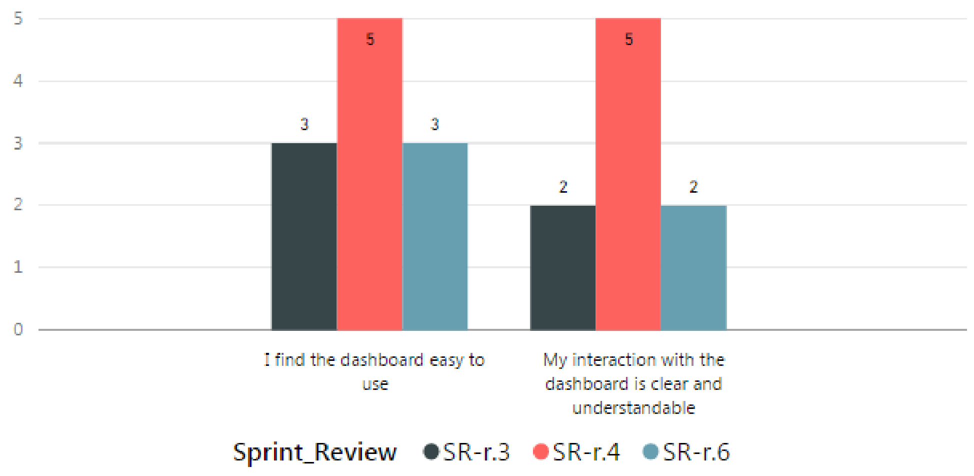


Figure 6.17: DocGen acceptance scores on Effort Expectancy

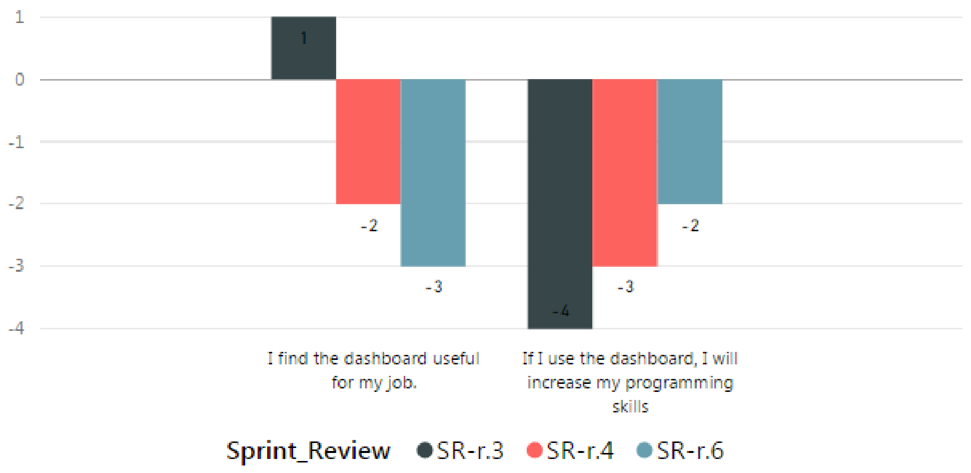


Figure 6.18: DocGen acceptance scores on Performance Expectancy

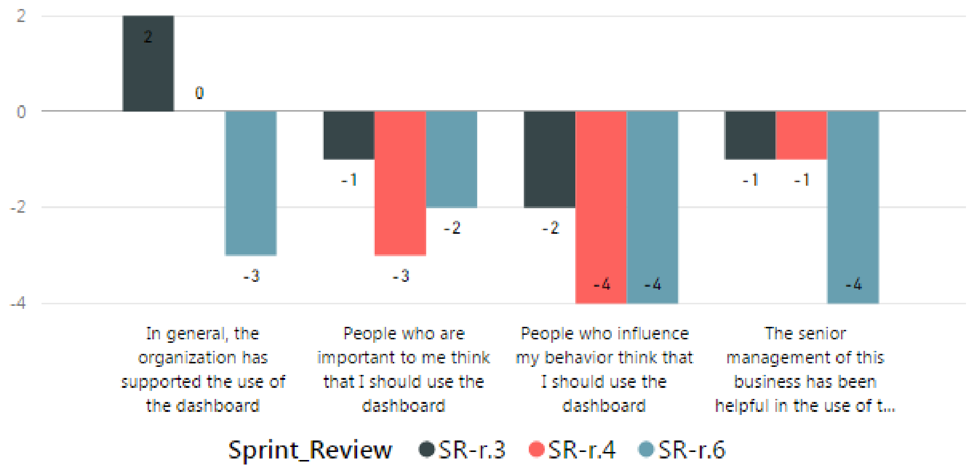


Figure 6.19: DocGen acceptance scores on Social Influence

Social Influence

Figure 6.19 presents the scores on the questions of *Social Influence*. The first statement (*In general, the organization has supported the use of the dashboard*), starts with a score of +2 at SR-r.3. This is the only positive value of the construct and it changed to score of 0 at SR-r.4 and -3 at SR-r.6. The second statement (*People who are important to me think that I should use the dashboard*), starts at -1, decreases to -3 at SR-r.4 but increases to -2 at SR-r.6. The third statement (*People who influence my behavior think that I should use the dashboard*), starts at -2, decreases to -4 at SR-r.4 and remain -4 at SR-r.6. The fourth statement (*The senior management of this business has been helpful in the use of the dashboard*), shows a score of -1 at the first two measurements but decreases to -4 at the end.

Chapter 7

Case RetailSystem

The second case of the embedded case study involves a transaction system called "Retail-System". The next sections start with an introduction of the software product and the development team. After the introduction follows the obtained survey results.

7.1 Introduction

Retail System is a commercial software product for retail stores (e.g. super markets) to connect Point Of Sale systems (i.e. cash registers) to management systems. It collects all the transactions and provides a complete overview of the transactions per store or region. It also allows the generation of special coupons for discounts. The main functionality is the processing of the transactions which can be at 80.000 per hour during the holidays. The product has a customer base of 110 customers in 30 countries, counting more than 20.000 stores and 75.000 points of sales which processes more than 20 billion transactions on annual basis.

7.1.1 Architecture

The system has different configurations depending on the size of the organization. Figure 7.1 provides an overview of the possible configurations of Retail System. On the left side of the figure is a single tier configuration with the application- and database on the same hardware. This configuration applies for small to medium sized enterprises (SME). The multi tier configuration provides multiple servers with a loadbalancer to save the transaction in a shared database. The last option is to use a multi-tier setup dedicated to high availability. This configuration only applies to large retail chains with an extreme

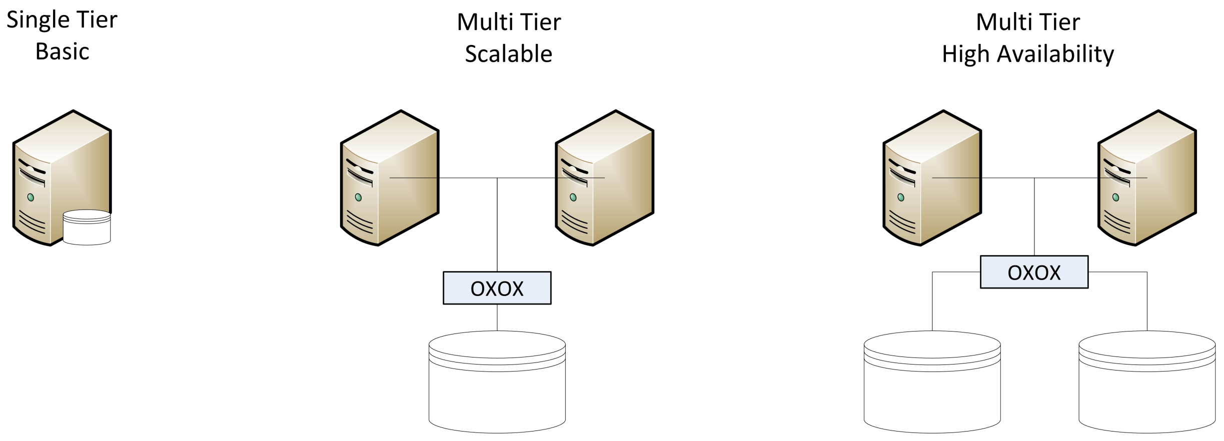


Figure 7.1: Configuration versions of RetailSystem

Table 7.1: Hardware configuration servers

Property	Application/Database Server
Brand/model	HP ML150 G5
Processor	Intel Xeon E5410 @ 2.33 GHz
Cores/Threads	4/4
FDB / TDP	1333 Mhz / 80 w
Chipset	Intel 5100
Memory	4096 MBytes FB-DDR2
OS	Windows Server 2012 R2 Standard 64-bit

peak load during the holidays. Our test system consists of a single tier configuration with shared resources for the application and database on a dedicated server.

The computational stack of the test system is presented in figure 7.2. The hardware consists of a dedicated server (i.e. table 7.1) with Windows Server 2012 SP1 as operating system (OS). On top of the OS runs Java as middleware to support the application server with an application instance for the application side. On the database side runs SQL Server 2012 with a SQL database and the Retail System schema.

The computational stack of Retail System has the additional layer *middleware* compared to DocGen. In total there are thirteen monitored processes. On the application server are two processes of *java* (32 and 64-bit), four worker processes spawned by IIS (*w3wp.exe*, and a service manager *nssm.exe*. On the database runs SQL Server 2012, with the processes *sqlserver.exe*, *sqlwriter*, *sqlbrowser.exe*, *sqlagent.exe*, *fdhost.exe*, *fdlauncher.exe*.

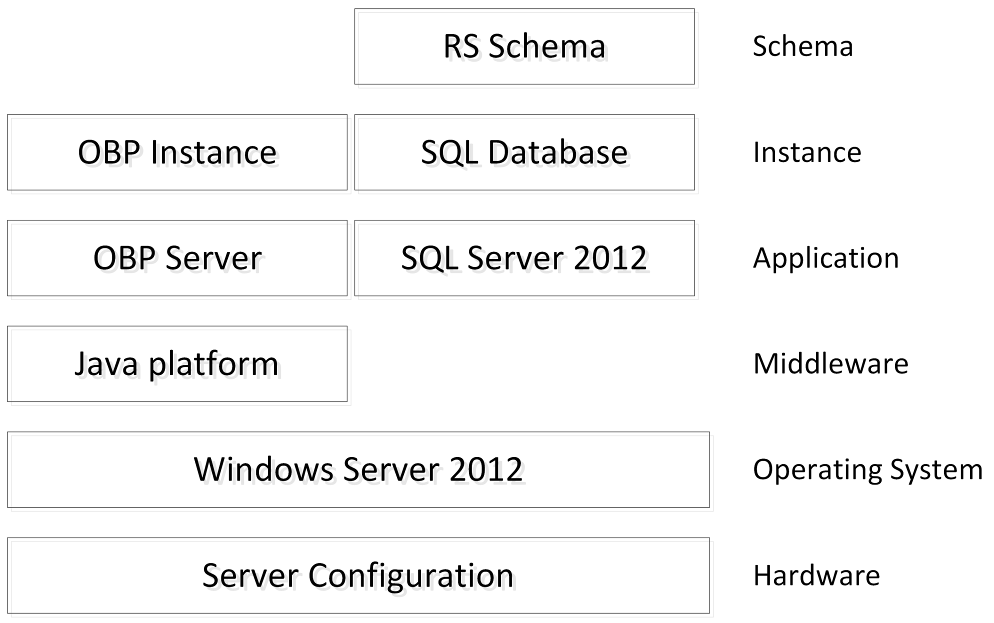


Figure 7.2: Computational stack RetailSystem showing application (left) and database (right)

7.1.2 The Development teams

The development team differs significantly from the first case. The product manager, product specialist and two development teams are located in one office in Belgium. Two other development teams are located in Romania. In total there are 23 stakeholders participating in our study: 16 developers, 1 database developer, 2 technical analysts, 2 testers, 1 software architect and 1 product specialist. The teams apply the principles of scrum with a three week sprint and a product release every two sprints.

7.1.3 Preparation

This subsection describes the preparation prior to the execution of the case study. The preparation of Retail System required more time due to geographical spread of the development teams. This situation required a different approach compared to the DocGen case.

Initial meeting

The initial meeting was with the product manager and product specialist of Retail System. During the meeting we gained commitment of the product manager and details about the application. Furthermore, we got in contact with the test engineer in Romania who performed the installation, upgrade and load test of the product on the server.

Test Environment

Retail System was installed on a dedicated server in an office environment in one of the branch offices. The VLAN of the server also hosted a load testing environment and multiple other servers used for research purposes.

The first step after the installation was the Joulemeter calibration on the server. This step introduced issues with the installation of Joulemeter. Despite the method to circumvent the Windows 7 check by Joulemeter, it was not possible to calibrate the program on Windows Server 2012. We solved this issue by installing and calibrating Joulemeter on a server with similar hardware, one extra hard disk and Windows Server 2008 R2. The temporary server introduced another issue regarding the .NET framework. Joulemeter requires .NET 3.5 and is unable to run on recent .NET versions. By installing .NET 3.5 we were able to calibrate Joulemeter on the temporary server and create a power model saved as a xml file. The base of this power model presented a higher value due to the additional hard disk. To overcome this issue we rebooted the server, let it run idle for half an hour and measured the average power consumption over a period of ten minutes. Finally the original base power of 149 W in the power model was replaced by the new average base power consumption of 138 w as provided by the Watts up Pro energy meter.

The second step was to setup Performance Monitor. In the DocGen case a dedicated server was used for Performance Monitor to collect data on a separated server. With Transys there is only one server available on site. Attempts to remotely monitor the server from another location failed due to firewalls separating the different company networks. As a result, the Performance Monitor ran on the server of the application itself. Our initial tests showed a small increase of resource utilization of 0.088% CPU utilization and 263 Kb memory. Additionally, the followup runs were logged with the same metrics across different versions to keep the protocol consistent.

The RetailSystem case provided one more issue with the setup of Joulemeter. The application RetailSystem has four IIS worker processes, each labeled as *w3wp.exe*. As a result, Joulemeter was only able to measure the energy consumption of the first worker process it found. The solution for this problem was to combine the process name with the process ID. To do so, we followed the steps from (Newton, 2010) to edit the registry and

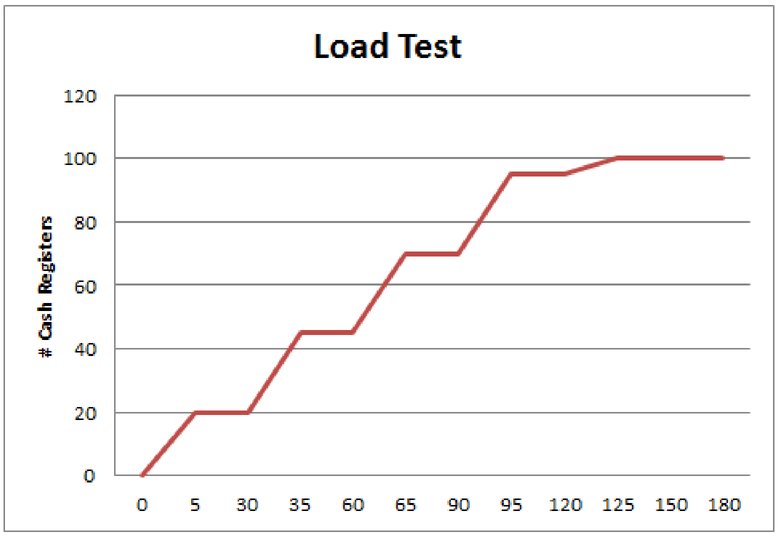


Figure 7.3: Loadtest of RetailSystem

allow Perfmon to combine the process name with the process, for example: *w3wp_1800*. By combining the process name and ID, we made unique labels which were accepted by Joulemeter.

Test Scenario

Retailsystem is used by retailers to log all the transactions of the cash registers during opening hours. The test scenario should be based on daily usage with low and high load on the server. Our test scenario involves a region manager with ten stores. During the afternoon the load rises until it reaches its max load. This corresponds with a usual shop scenario in which the number of visitors keeps rising during the afternoon and peaks around half an hour before closing time.

Our scenario starts with twenty cash registers, each conducting fifty transactions per hour. After every thirty minutes twenty-five cash registers are added within a five minute period, until a total of one hundred cash registers, each conducting fifty transactions per hour, are reached. Figure 7.3 provides a graph of the number of cash registers on the Y-axis and the time expressed in minutes on the X-axis.

7.1.4 Execution

This section describes the execution of the case Retailsystem. The execution protocol was five times executed. The first round did not capture the awareness as it was necessary to create a baseline of the software product.

Installation

The installation process of RetailSystem is automated with the help of a release manager. The updates were performed by the test engineer due to missing permissions to access the test environment.

Execution of the test

The execution of the test was exactly the same as described in the case study protocol. The load tests were activated by the test engineer or product specialist at certain times a day. Each consisted of half an hour preparation, three hours of testing and half an hour of stopping the test and collecting all of the data. The target was to get ten valid test runs for every version. Due to various reasons we ended up with 9 valid runs for the first two releases.

ETL

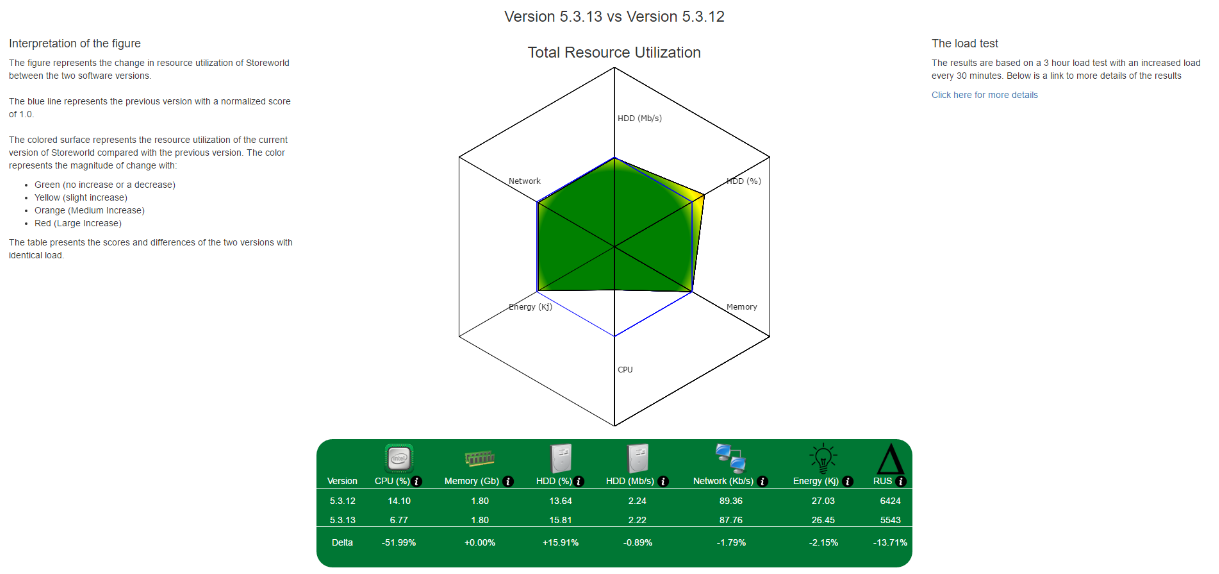
The log files were copied to the workstation and converted to become readable .csv files for Excel. The data from the files was loaded in the workspace file and reduced to the minimum required data. This step reduced the the size of the data by 70%.

Data Analysis

The data analysis encompassed the calculation of the metrics. Each load test was analyzed to determine a validity. A test was invalid if the data contained an extreme outlier which could not be explained. The calculated metrics were the same as described in the case study protocol without the metric *execution time*. The test scenario of Retail System had a fixed execution time of three hours. Thus, the metric execution time was not suitable to apply.

Dashboard

The dashboard of RetailSystem consists of two pages. The first page (7.4) is the total resource utilization of Retail System on the server, together with a table presenting all of the details. The main page also presented an explanation on the metrics, the executed



Interpretation of the figure

The figure represents the change in resource utilization of Storeworld between the two software versions.

The blue line represents the previous version with a normalized score of 1.0.

The colored surface represents the resource utilization of the current version of Storeworld compared with the previous version. The color represents the magnitude of change with:

- Green (no increase or a decrease)
- Yellow (slight increase)
- Orange (Medium Increase)
- Red (Large Increase)

The table presents the scores and differences of the two versions with identical load.

The load test

The results are based on a 3 hour load test with an increased load every 30 minutes. Below is a link to more details of the results

Click here for more details

Version	CPU (%)	Memory (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 7.4: RetailSystem dashboard total resource utilization

test and a hyperlink to the second page. The second page (7.5)showed six smaller graphs with corresponding tables. Each graph presented the resource utilization of thirty minutes and the specific load (number of users).

7.1.5 Survey

The survey was held with an online form due to the geographical distance. The disadvantage of this approach was the lower participation rate as participants could easily ignore the survey sent to them.

7.2 Results RetailSystem

The next subsections describe the results of the RetailSystem case. The first analysis presents the results of the awareness before and after the introduction of the energy dashboard based on survey A and B with sprint reviews SR-r.1 and SR-r.6. The second analysis presents the development of the awareness, based on survey B, among all the sprint reviews. The third analysis presents the user acceptance based on survey C.

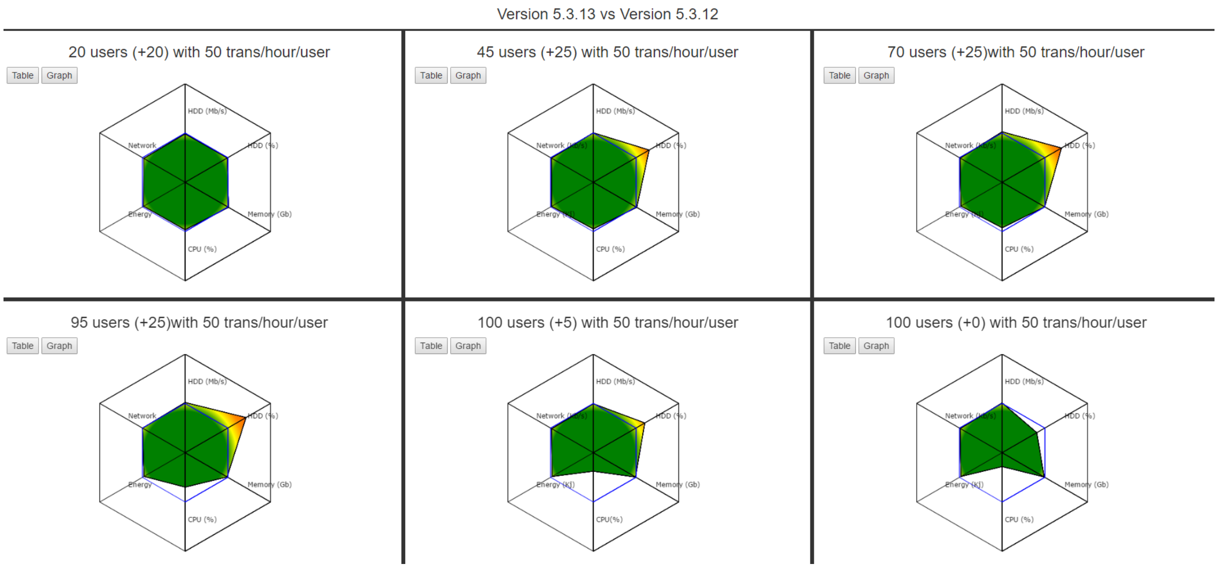

Version 5.3.13 vs Version 5.3.12
20 users (+20) with 50 trans/hour/user
45 users (+25) with 50 trans/hour/user
70 users (+25)with 50 trans/hour/user
95 users (+25)with 50 trans/hour/user
100 users (+5) with 50 trans/hour/user
100 users (+0) with 50 trans/hour/user
Table
Graph
HDD (Mb/s)
HDD (%)
Memory (Gb)
CPU (%)
Energy
Network

Figure 7.5: RetailSystem dashboard with details of scenarios

7.2.1 Awareness before & after

The initial measurement of SR-r.1 provided the data to create the awareness baseline of the participants. The baseline scores can be compared with the scores of SR-r.6 to determine the change in awareness before and after the introduction of the energy dashboard. The initial score, based on the positive statements, was +4 which decreased to -16 at the last sprint review. Figure 7.6 provides an overview of the scores per awareness construct at the start (SR-r.1) and end (SR-r.6) of the case study.

C1 *Consciousness of the energy consumption of software* starts with -13 and decreases to -18 at the end of the case study. C2 *Consciousness of the relevance of creating energy efficient software* starts with +4 and ends with -2. C3 *Consciousness of one's possibilities to reduce the SEC* starts and ends at +8. C4 *Personal Opinion on reducing the SEC during software development* changes from +4 at the start to -7 at the end of the case study. C5 *Social norm on developing energy efficient software* changes from -7 to +1. C6 *Other motives to reduce the SEC* decreases from +8 to +2 at the end of the case study.

C1 - Consciousness of the energy consumption of software

Figure 7.7 presents the scores of C1 per statement. The first question (*Before this project I wondered multiple times about the energy consumption of our software development environment*), starts with -12. At the end of the case study the score decreased to -16.

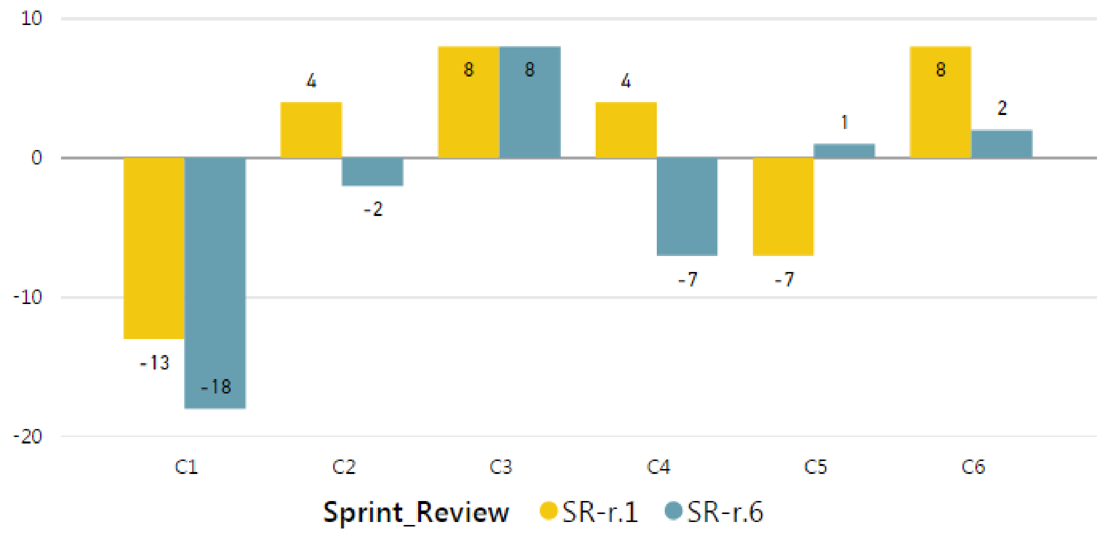


Figure 7.6: The scores per awareness construct before and after the introduction of the energy dashboard

The second question of C1 (*I want to determine the energy consumption of our software development environment (e.g. ..)*) started with -1. At the end of the case study the score has become -2.

C2 - Consciousness of the relevance of creating energy efficient software

The scores on the statements of C2 are presented in figure 7.8. The first question on the left (*I expect that software has a large influence on the energy usage*) starts with +1 at SR-r.1 and ends with -6 at SR-r.6. The second question of C2 (*I would like to know the energy consumption of our software product*) starts with +3, which into +4 at the end of the case study.

C3 - Consciousness of one's possibilities to reduce the SEC

The scores per statement of C3 are presented in figure 7.9. The first statement (*If I had more time to work on the code, I would be able to reduce the energy consumption*) starts with -1 and ends with -2 at SR-r.6. The second statement (*It is possible to make a trade-off between our current software qualities (..) and the energy consumption.*) starts at +5 and ends with +4. The third statement (*The applied techniques (..) to realize the software product allow for the reduction of the energy consumption*) starts with +4 and ends with +6.

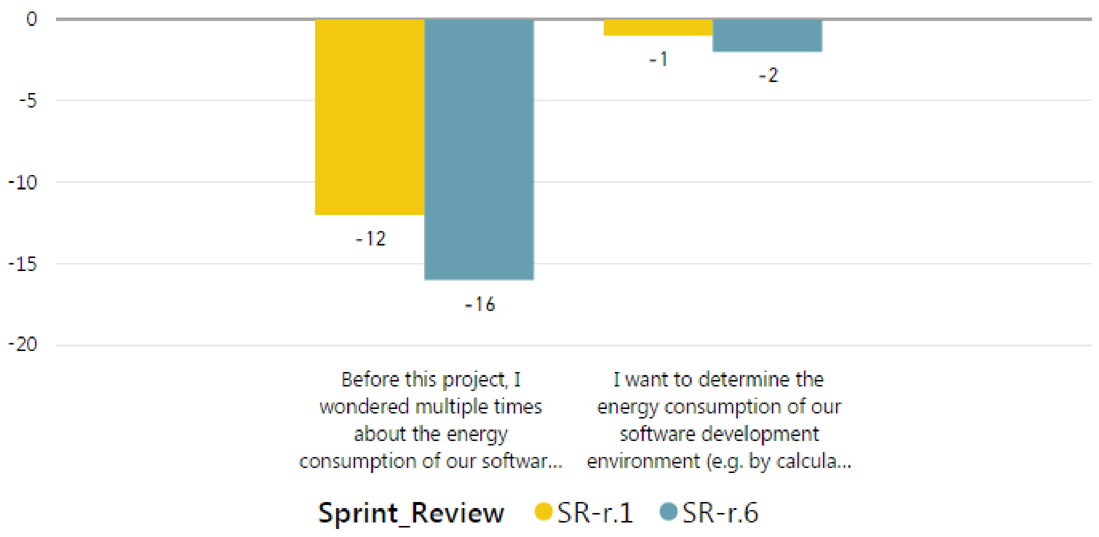


Figure 7.7: RetailSystem awareness scores per statement of C1

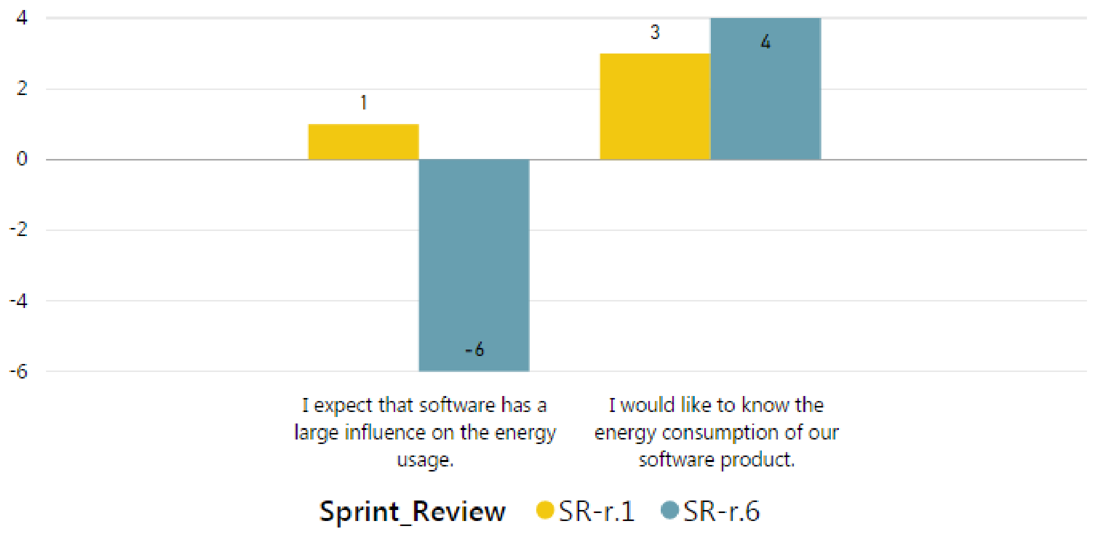


Figure 7.8: RetailSystem awareness scores per statement of C2

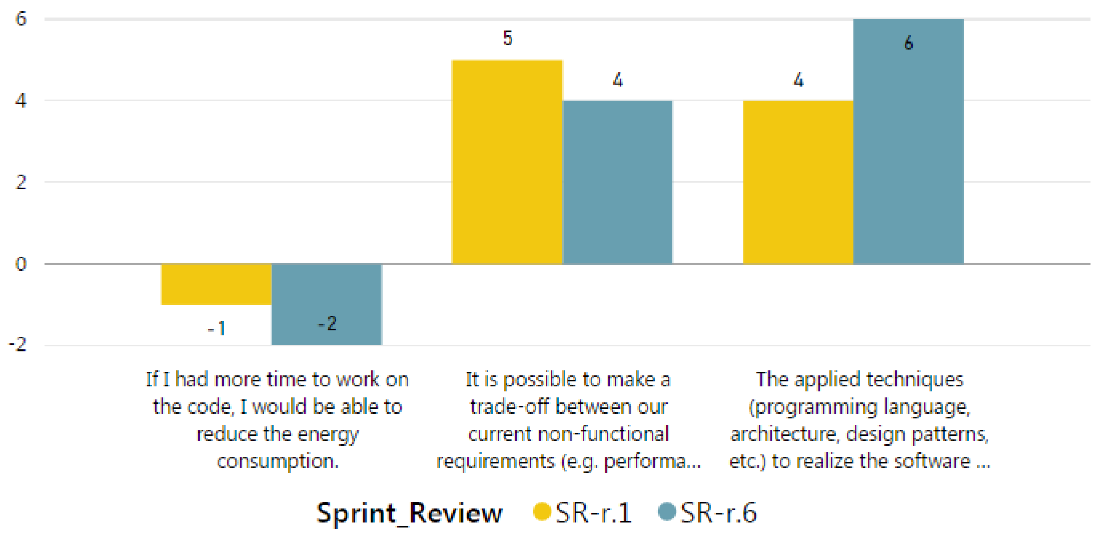


Figure 7.9: RetailSystem awareness scores per question of C3

	SR-r.1	SR-r.6
Developers	0	0
Software Architects	5	5
Developers + Software Architects	6	6
Other, namely...	Hardware Manufacturers, Testers	Hardware Manufacturers, Testers

Table 7.2: RetailSystem distribution of answers

Besides the statements answered with the Likert-items was the question *The energy consumption of software should be addressed by:* with the options *Developers*, *Software Architects*, *Developers and Software Architects* and *Other, namely...*. The results of this question are presented in table 7.2. The output answers at SR-r.1 and SR-r.6 are identical.

C4 - Personal Opinion on reducing the SEC during software development

The scores per statement of C4 are presented in figure 7.10. The first statement (*Addressing the energy consumption of our software should gain more attention*) starts with +1 at SR-r.1 and decreases -1 at SR-r.6. The second statement (*I would like to reduce the energy consumption, if I am allowed to spend time on it*) starts at +3 and has become -6 at the end of the case study.

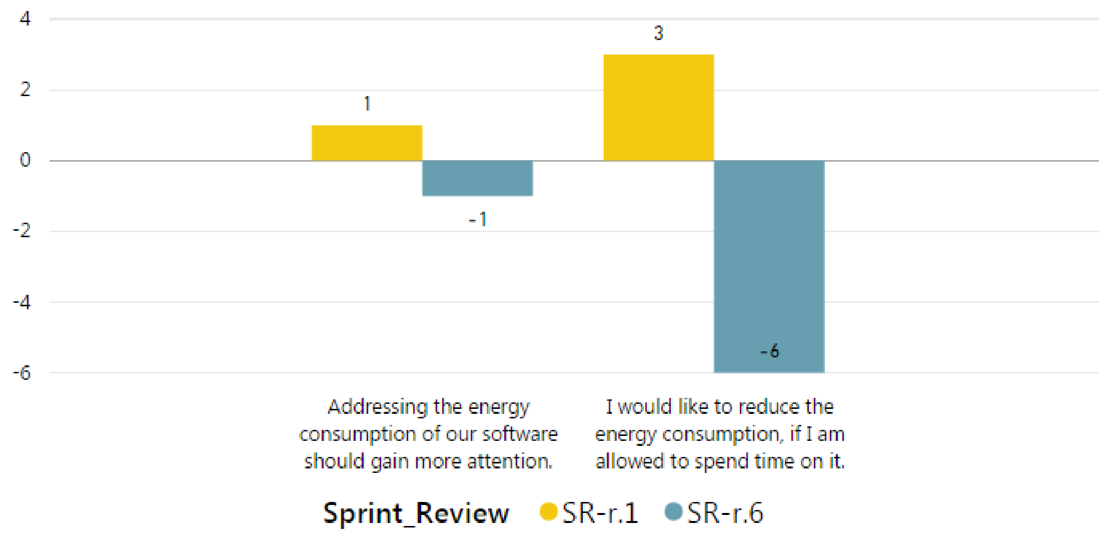


Figure 7.10: RetailSystem awareness scores per statement of C4

C5 - Social Norm (..) on developing energy efficient software

Figure 7.11 presents the scores per statement of C5. The first statement (*If other teams reduce the energy consumption of their software, I would attempt it too*) starts with +1 at SR-r.1 and ends with +3 at SR-r.6. The second statement (*Reducing the energy consumption would be a benefit to the SPO and to the customer*) starts with +9 and ends with +6. The third statement (*The energy consumption of our software product is discussed during (in)formal meetings*) changes from -17 to -8.

C6 - Other motives to reduce the SEC

Figure 7.12 presents the scores per statement of C6. The first statement (*Code optimization to improve non-functional requirements should be acknowledged an included in our backlog*) changes from +11 to +6. The second statement (*The benefits of rewriting the code to reduce the energy exceed the costs*), decreased from -3 to -4.

7.2.2 Development over time

Constructs C2 and C3 were measured at every sprint review with the exception of SR-r.2. The development in scores of these two constructs are able to provide more details of the development between the start and end of the case study. Figure 7.13 presents the scores

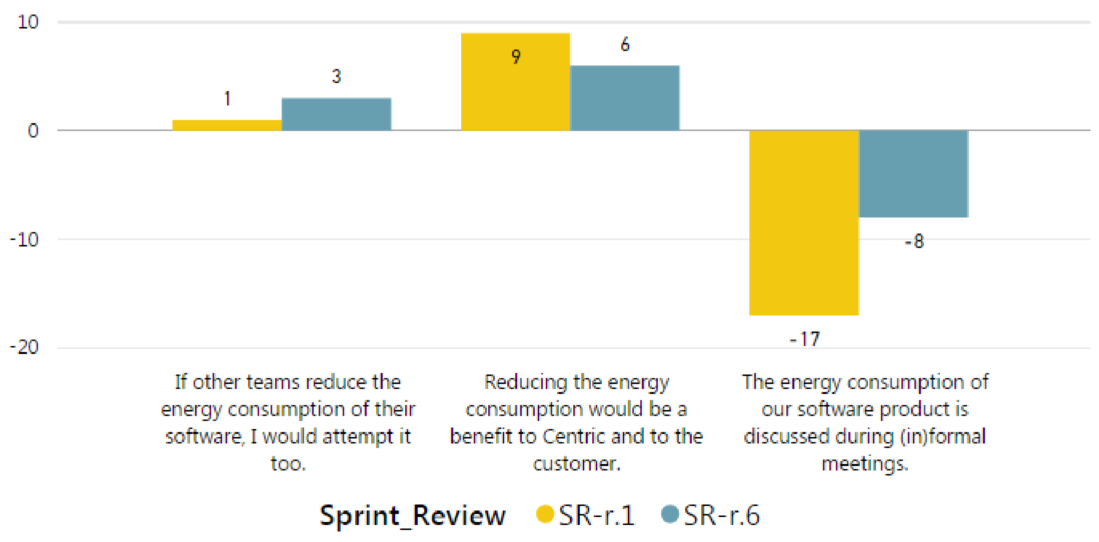


Figure 7.11: RetailSystem awareness scores per statement of C5

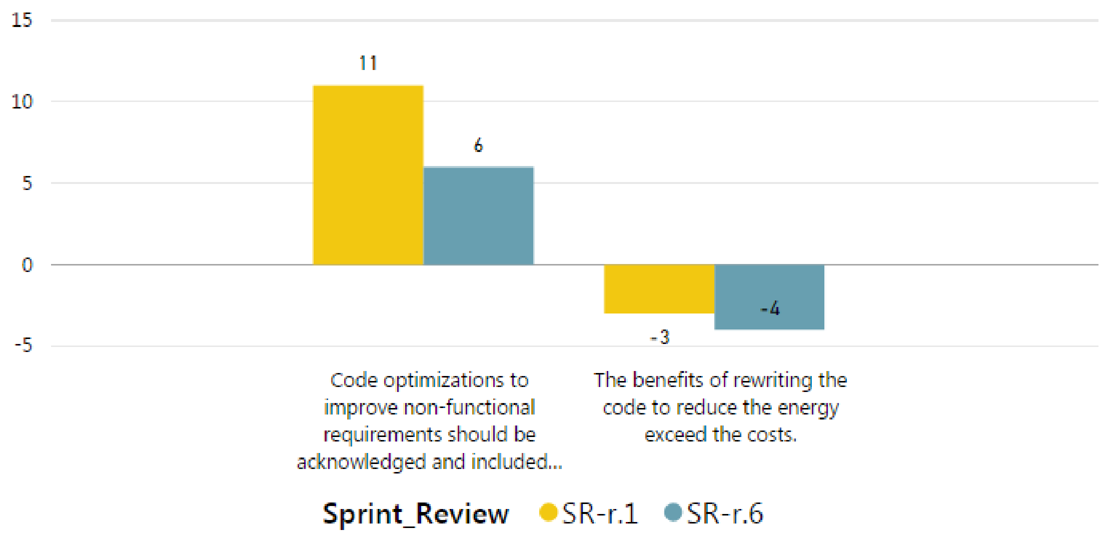


Figure 7.12: RetailSystem awareness scores per statement of C6

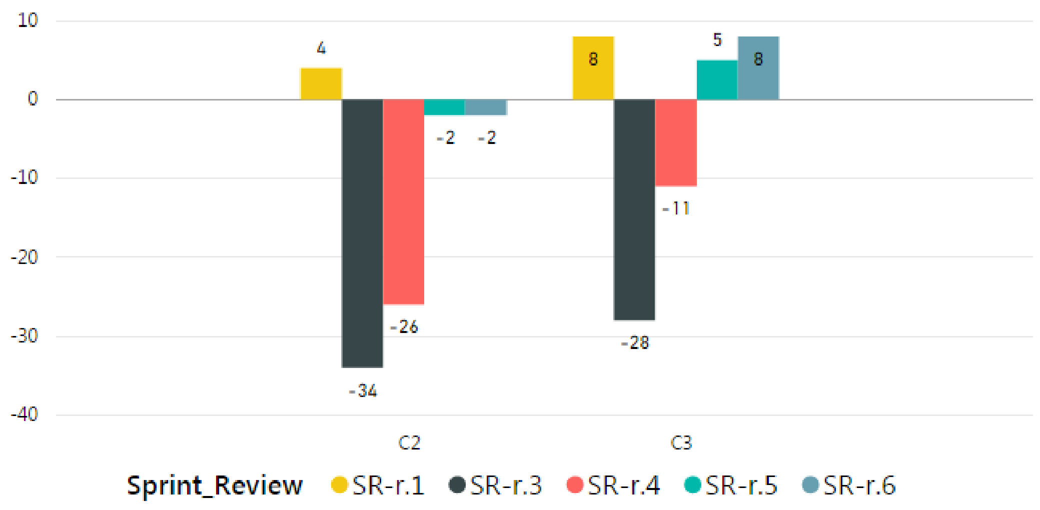


Figure 7.13: RetailSystem acceptance scores of C2 and C3 over the sprint reviews

of the two constructs over the five measured sprint reviews. The scores of C2 start with +4 at SR-r.1 and decrease to -34 at SR-r.3. At SR-r.4, the score increases to -26 and -2 at SR-r.5. The last score at SR-r.6 remains identical at -2. The scores of C3 follow a similar development. At the start the case study, the score of C3 is +8 which decreases at SR-r.3 to -28. At SR-r.4, SR-r.5 and SR-r.6 the score increases to -11, +5 and +6 respectively.

Figure 7.14 presents the scores per statement of C2 and C3. The first statement (*I expect that software has a large influence on the energy usage*), starts with +1 at SR-r.1 but decreases to -15 at SR-r.3. At SR-r.4 and SR-r.5, the scores increase to -11 and -5 but decrease again to -6 at SR-r.6. The second statement (*I would like to know the energy consumption of our software product*), starts at +3 and decreases to -19 at SR-r.3. In the successive sprint reviews, the score increases to -15, +3 and +4. The third statement (*If I had more time to work on the code, I would be able to reduce the energy consumption*), starts with -1 followed by -9, -4, -1 and -2. The fourth statement, (*It is possible to make a trade-off between our current non-functional requirements and the energy consumption*), starts with +5 at SR-r.1. At SR-r.3 the score decreases to -6 and increases at SR-r.4, SR-r.5 and SR-r.6 to 0, +3 and +4. The last statement, (*The applied techniques (...) allow for the reduction of the energy consumption*), starts at +4. At SR-r.3 the score decreases to -13 but increases in every successive sprint to -7 (SR-r.4), +3 (SR-r.5) and +6 (SR-r.6).

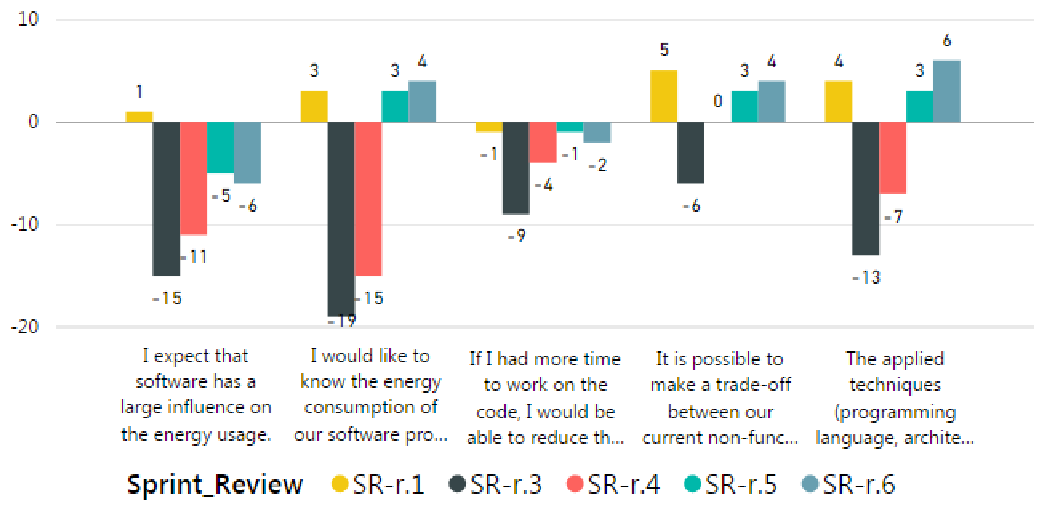


Figure 7.14: RetailSystem awareness scores of the statements of C2 and C3 among all sprint reviews

7.2.3 User Acceptance

The change of user acceptance was collected with survey C during the sprint review meetings. There were four measurements at sprint review SR-r.3, SR-r.4, SR-r.5 and SR-r.6. Figure 7.15 presents the acceptance scores per construct. AT, *Attitude towards using technology* starts with -13 at SR-r.3 and increases to -10 at SR-r.4. At SR-r.5 and SR-r.6 the scores decrease to -14 and -16 respectively. BI, *Behavioral Intention* starts with a score of -25 at SR-r.3. At SR-r.4 and SR-r.5, the score drops to -34 and -39. Finally, at SR-r.6 the score increases again to -30. EE, *Effort Expectancy*, starts with -4 at SR-r.3 and increases to -2 for SR-r.4 and SR-r.5. At SR-r.6 the score increases again to +7. PE, *Performance Expectancy*, starts at -10 and decreases to -14 at SR-r.4. At SR-r.5 and SR-r.6 the scores increase again to -11 and -10. SI, *Social Influence*, starts with -37 at SR-r.3. In the three successive sprint reviews the score increases to -36 (SR-r.4), decreases again to -37 (SR-r.5) and increases again to -36 (SR-r.6).

The next subsections present the constructs with the related statements. The scores of each statement need to put into perspective for correct interpretation. The minimum and maximum score are -24 and +24 with 0 as the neutral center. The change of one person changing its opinion by one option already causes a change of one point in the total score. Therefore a change of one or two points does not indicate a significant change among all the participants.

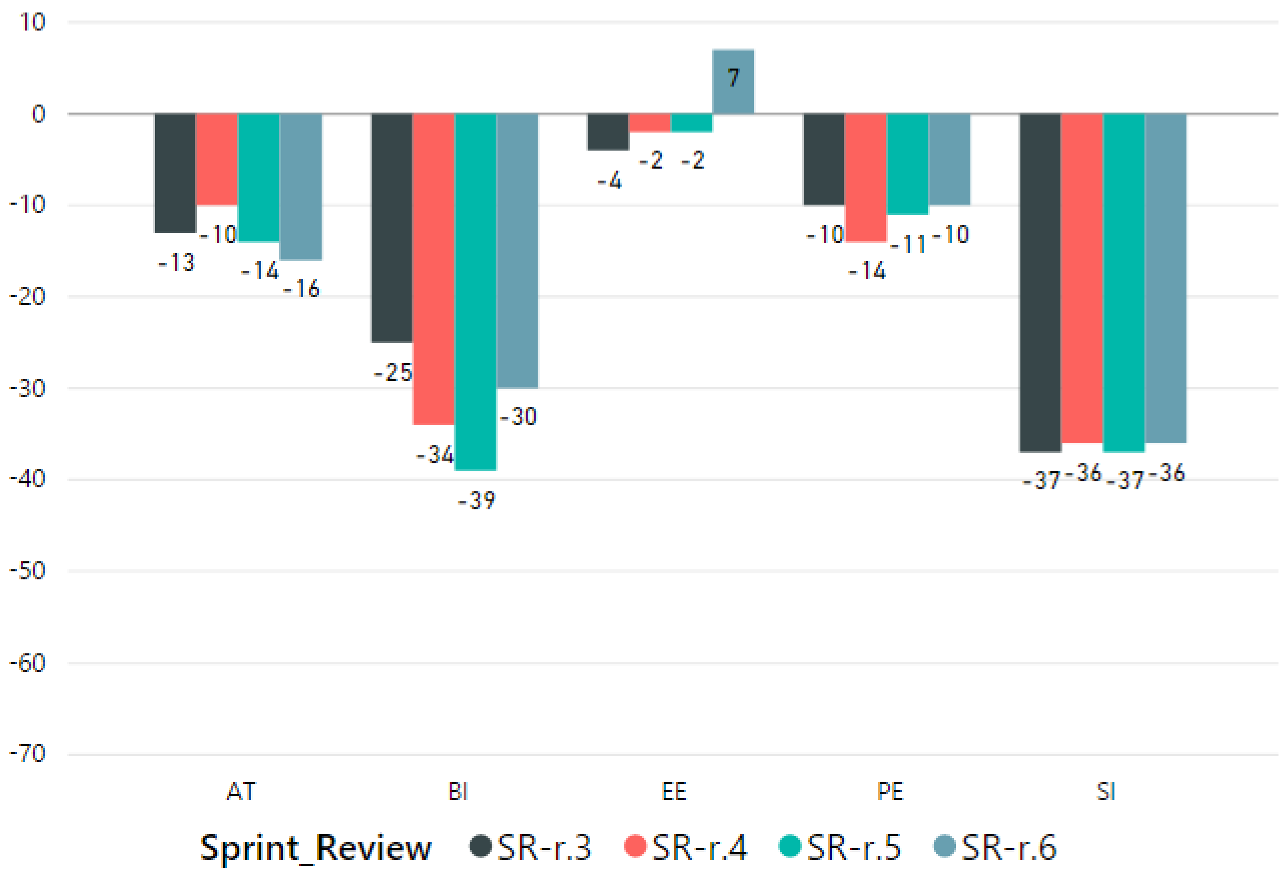


Figure 7.15: RetailSystem acceptance scores per construct among the sprint reviews of survey C

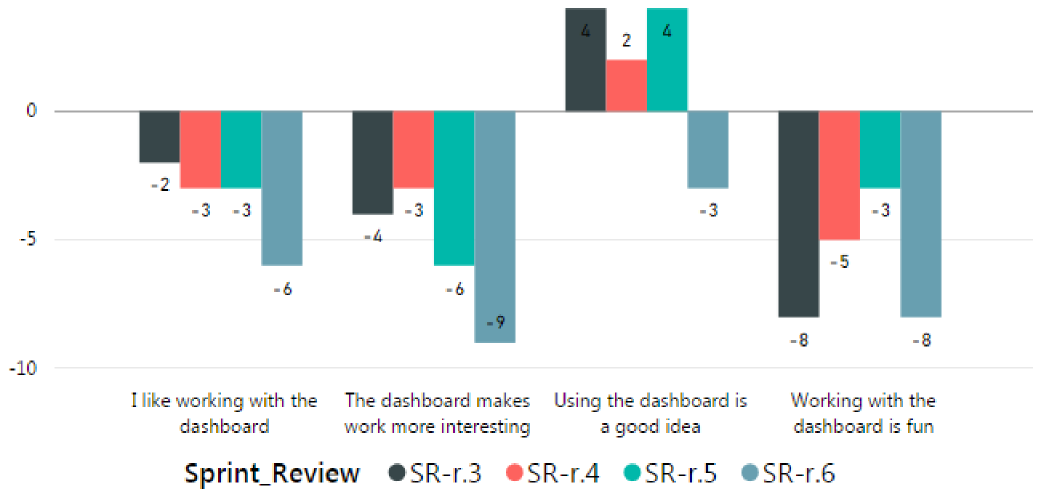


Figure 7.16: RetailSystem acceptance scores on Attitude towards using technology over the sprint reviews.

Attitude towards using technology

Figure 7.16 presents the scores on the statements of AT. A general trend among the four statements is the decrease in scores over time. The first statement (*I like working with the dashboard*) starts with a score of -2 at SR-r.3 which decreases to -3 at SR-r.4 and SR-r.5 and has a decline to -6 at SR-r.6. The second statement (*The dashboard makes work more interesting*), starts with -6 and increases to -3 at SR-r.4. At SR-r.5 and SR-r.6 the scores drop to -6 and -9. The third statement (*Using the dashboard is a good idea*) has a completely different pattern. The score at SR-r.3 starts with +4, decreases to -2 at SR-r.4, increases again to +4 at SR-r.5 and decreases to -3 at SR-r.6. Finally, the fourth question (*Working with the dashboard is fun*) starts with a score of -8. In the two successive sprints increases to -5 and -3 but decreases again to -8 at SR-r.6.

Behavioral Intention

The scores on BI over the sprint reviews are shown in figure 7.17. The construct BI consists of three questions. Visual inspection of the graph shows a general negative sentiment represented by predominantly negative scores. The first question (*I intend to use the dashboard in the next sprint*) starts with a score of -3. In the successive sprint this score decreases to -6, -7 and -11. The second question (*I plan to use the dashboard in the next sprint*) starts with -10 at SR-r.3, increases to -8 for SR-r.4 and SR-r.5, to decrease again to -10 at SR-r.6. The third question (*I predict I will use the dashboard in the next*

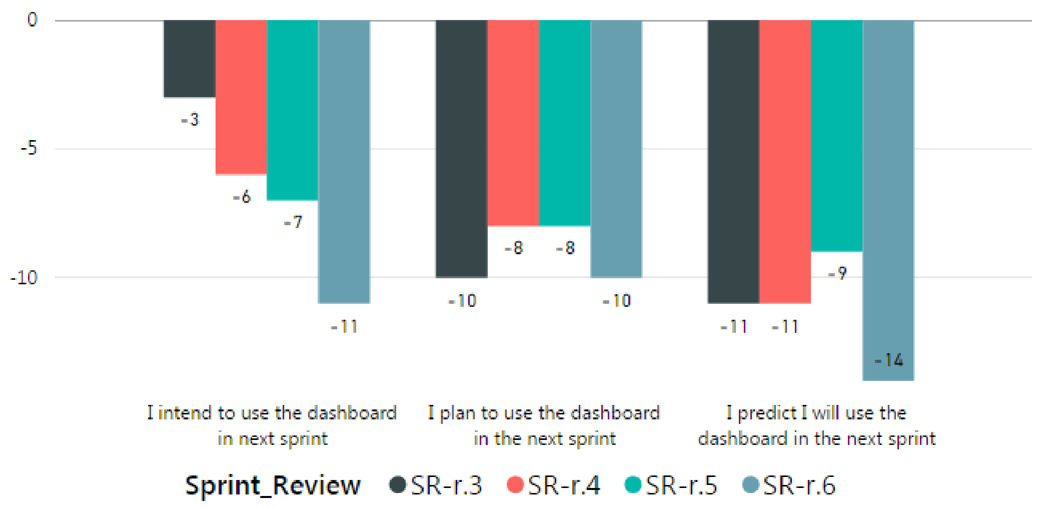


Figure 7.17: RetailSystem acceptance scores on Behavioral Intention over the sprint reviews.

sprint) starts with -11 at SR-r.3 and SR-r.4, increases to -9 at SR-r.5 and decreases again to -14.

Effort Expectancy

The scores on the questions of EE are presented in figure 7.18. The first question (*I find the dashboard easy to use*) starts with a score of +1 which increases to +3 and +7 in the two successive sprints and decreases to +4 at the last sprint review. A similar trend occurs at the second question (*My interaction with the dashboard is clear and understandable*) with scores -4, -1, +4 and +2.

Performance Expectancy

Figure 7.19 presents the scores on PE over the sprint reviews. The first statement (*I find the dashboard useful for my job*) starts with a score of +2 at SR-r.3. In the successive sprints the score decreases to -2 (SR-r.4), 0 (SR-r.5) and -4 (SR-r.6). The second statement (*If I use the dashboard, I will increase my programming skills*) has a different trend with increasing scores at SR-r.4 and SR-r.5 and again a strong decrease at SR-r.6.

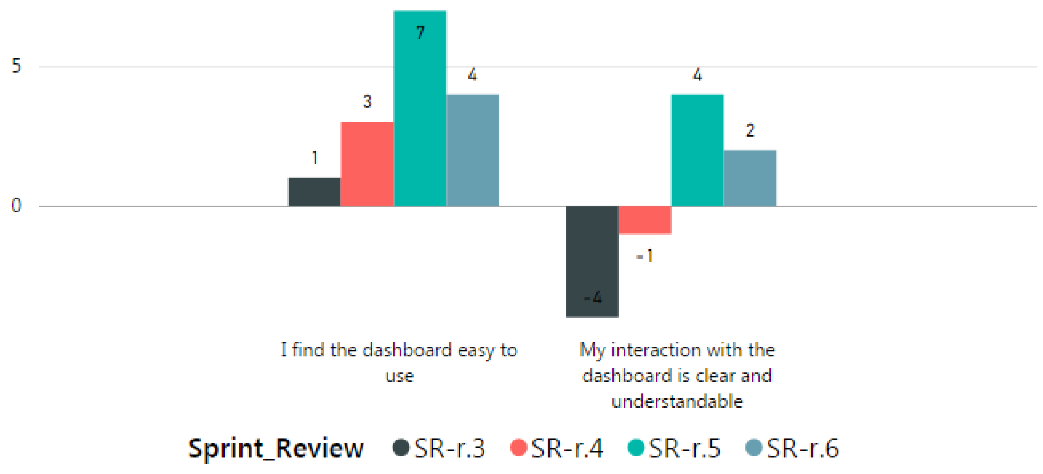


Figure 7.18: RetailSystem acceptance scores on Effort Expectancy over the sprint reviews.

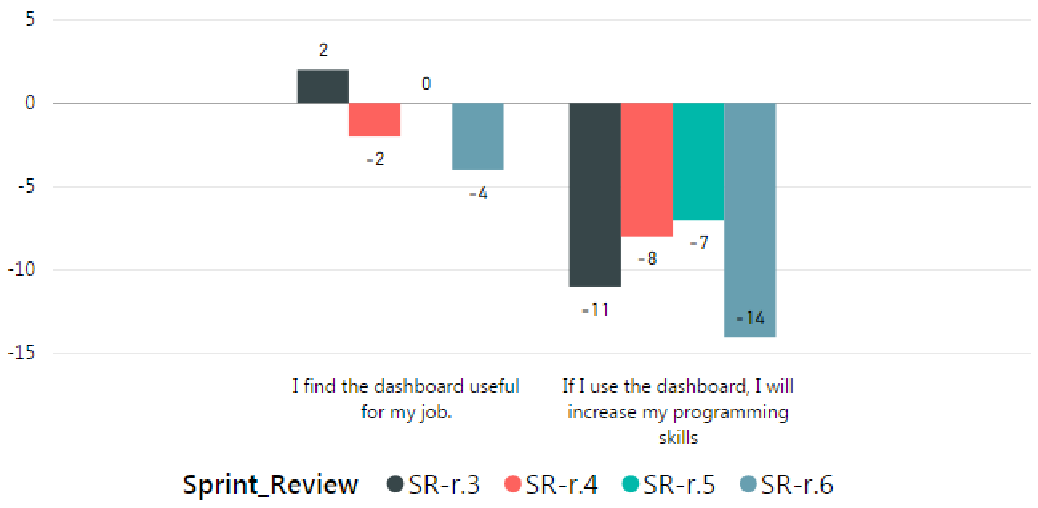


Figure 7.19: RetailSystem acceptance scores on Performance Expectancy over the sprint reviews.

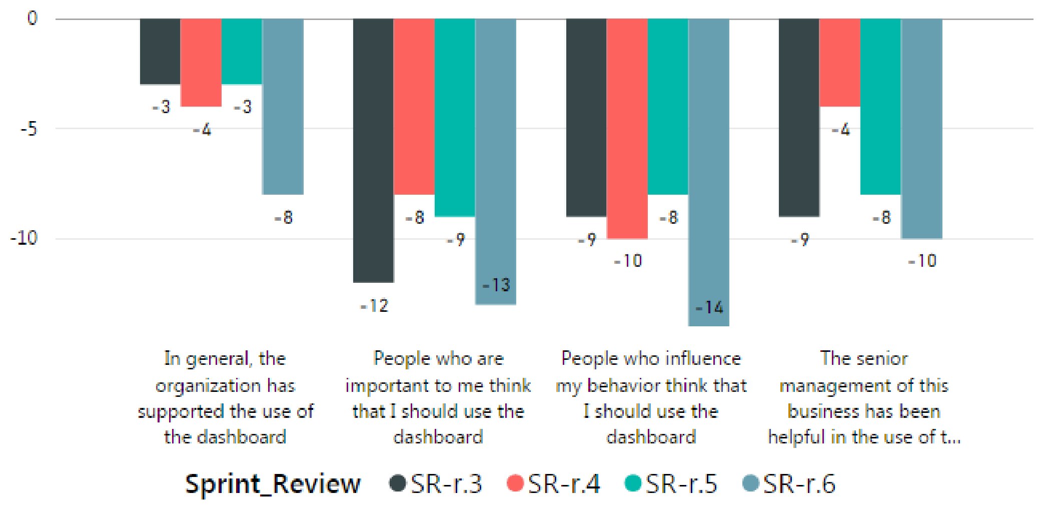


Figure 7.20: RetailSystem acceptance scores on Social Influence over the sprint reviews.

Social Influence

Figure 7.20 presents the scores on SI over the sprint reviews. The first statement (*In general, the organization has supported the use of the dashboard*) contains neutral values of -3 (SR-r.3), -4 (SR-r.4) and -3 (SR-r.5) and a negative value of -8 (SR-r.6). The second statement (*People who are important to me think that I should use the dashboard*) starts with a score of -12. At SR-r.4 the score increases to -8 and decreases at SR-r.5 again to -9. Finally, the score drops again at SR-r.6 to -13. The third statement (*People who influence my behavior think that I should use the dashboard*) starts with -9 which fluctuate at SR-r.4 and SR-r.5 with -10 and -9. The final score at SR-r.6 decreases to -14. The fourth and final statement (*The senior management of this business has been helpful in the use of the dashboard*) starts with a negative score of -9 which increases to -6 at SR-r.4 to decrease again to -8 at SR-r.5 and -10 at SR-r.6.

Chapter 8

Discussion

In this chapter we discuss the results from the two cases. The chapter starts with the main topic of this thesis: creating awareness on the energy consumption of software. After this discussion follow different findings based on analysis of both cases. Finally, we conclude this chapter with the limitations of our research.

8.1 Creating Awareness

The background literature on awareness presented the model on changing environmental detrimental habits. We applied this model in the context of green software and quantified each construct by means of survey A and B. A change in score of a construct indicates a change in consciousness and therefore a change in awareness.

If we look at the development of the construct scores of both cases, we find more decreasing scores, than increasing scores. For DocGen, C1 (*consciousness of the energy consumption of software*) and C5 (*social norm*) increase in score, while the other constructs decrease in score. For RetailSystem, only C5 increases at construct level while C3 (*consciousness of one's possibilities to reduce the SEC*) remains equal and the other constructs decrease in score.

Survey B, with C2 (*consciousness of the relevence on creating energy efficient software*) and C3 (*Consciousness of one's possibilities to reduce the SEC*), provides more detail on the development of the scores over the sprint reviews. Both cases have a strong decrease in scores of both constructs after they are confronted with the SEC of their software product. However, these score start increasing again after a few sprints. For the case of RetailSystem, this increasing trend starts at SR-r.4. The participants of RetailSystem report neutral scores at SR-r.5 and even positive scores on C3 (*consciousness of one's* *possibilities to reduce the SEC*) at SR-r.6. The increasing scores of the participants of DocGen start at SR-r.6. The ongoing decline of DocGen might have been caused by the absence of 2 participants at SR-r.3 with the result of a lower sum of scores at SR-r.3. The participants of DocGen report neutral scores again at SR-r.6.

In conclusion, the participants from both cases report to be neutral or positive at the start of the case study about green software. They become strongly negative after they are confronted with the implications of green software. However, at the end of the case study, they are able to form a weighted decision about the position of the SEC in software engineering. In other words, the participants have become aware of green software.

8.2 Knowledge Gap

A closer look at the results among the statements reveals a knowledge gap among the participants.

The stakeholders from DocGen provide neutral scores, when asked whether the energy consumption of their software product could be reduced given more time. In addition, they are also neutral about making a trade-off between other software qualities and the software energy consumption. Finally, the scores of the stakeholders at the end of the case study also indicate a neutral attitude when asked whether they see possibilities of positioning the SEC within the applied techniques (architecture, programming language, design patterns, etc.). To summarize, the stakeholders of DocGen have become neutral about three things: 1) addressing the SEC even when given more time, 2)about making the trade-off with with other software qualities and 3) about the possibilities within the applied techniques.

The stakeholders of RetailSystem provide negative scores, when asked whether the energy consumption of their software product could be reduced given more time. In other words, even when they would get more time to address the SEC, they wouldn't be able to accomplish it. However, the stakeholders of RetailSystem are neutral about making a trade-off between other software qualities and the SEC. In contrast, the stakeholder of RetailSystem do see possibilities of positioning the SEC within the applied techniques (architecture, programming language, design patterns, etc.) at the end of the case study. To summarize, the stakeholders of RetailSystem are negative about 1) addressing the SEC even when given more time, 2) neutral about making a trade-off between other software qualities and the SEC, 3) positive about the possibilities of positioning the SEC within the applied techniques (architecture, programming language, design patterns, etc.) at the end of the case study.

In addition, we asked participants from both cases which role(s) should be responsible for addressing the SEC. All of the DocGen stakeholders report 'software architects and developers', with one addition of 'testers'. The addition of testers is a logical outcome as one of the participants was a tester while the others were all developers. The stakeholders of RetailSystem provide multiple answers. Half of the stakeholders of RetailSystem answer the combination 'software architects and developers', the other half reports solely 'software architects', in addition the role 'hardware manufacturers' were multiple times mentioned.

The answers from both cases on who should be responsible for addressing the SEC, explain the previous results. The participants of DocGen have become neutral about addressing the SEC, positioning the SEC and the possibilities to address the SEC while they do acknowledge the SEC as their own responsibility. Therefore, the neutral sentiment towards the SEC, is a result of not knowing how to address the SEC, position it between other software qualities or identify possibilities in the current applied techniques.

The answers of the participants of RetailSystem also indicate a lack of knowledge, as they are unaware of how they should be able to reduce the SEC or position it between the other software qualities. However, there is a difference as half of the team consider it as their responsibility ('software architects and developers'), while the other half considers it as solely a task for the 'software architects' with the addition of 'hardware manufacturers'. This explains why they are positive about the possibilities, as they believe it is a task for others who should be more knowledgeable about it.

In conclusion, the participants from both cases acknowledge the SEC but are uncertain about bringing the concept of the SEC into a daily practice, indicating a knowledge gap among the development teams of both software products. This finding aligns with the feedback we received from both cases. Among the DocGen case there was one developer who was admitted he didn't know how to address the SEC and therefore was a bit frustrated about the whole project. Our contact at the RetailSystem case answered something similar as they find it interesting, but at the same time wonder how they are supposed to address the SEC.

8.3 Acceptance of the Energy Dashboard

The scores on survey C among both cases show an increasing negative sentiment among the constructs as the number of sprint reviews increase and suggest that the energy dashboard does not appear to be the right means to stimulate awareness on software energy consumption. However, the negative scores can be explained with the help of the previous results.

Looking at the results of acceptance in combination of the awareness results, we can explain the rejection of the energy dashboard. The awareness scores dropped strongly at the second presentation of the energy dashboard, as the participants were confronted with the results. The second sprint review meeting was also the first measurement of the acceptance and therefore could have caused the negative sentiment towards the energy dashboard. In conclusion, the confrontation with the software energy consumption of the software product caused a rejection towards the energy dashboard.

At the same time, the participants do seem to acknowledge the energy dashboard. Both cases indicate that the dashboard is easy to use, as shown by the positive scores at *Effort Expectancy.* Another positive aspect are the scores on the statement 'it is a good idea to use the dashboard', with both cases providing positive scores until the last sprint review meeting. At the last sprint review meeting most of the scores drop and become negative for both cases. This development might be caused by growing tired of being confronted with the software energy consumption, in combination with the inability to reduce the SEC. Therefore, it might be necessary to review the frequency of presenting the energy dashboard. Instead of being confronted every sprint review, it would make sense to adjust the frequency and only present it at times when there is major update. For example, the differences between release 8.0.7 and 8.0.7.1 of DocGen were minimal and therefore resulted in hardly any change in the software energy consumption.

Looking at the scores of *Behavioral Intention*, we notice a large difference between the two cases. The participants of DocGen start with a positive score at the start of the case study, but become more neutral and negative at the end of the case study. In contrast, the participants of RetailSystem already start with a negative score on *behavioral intention* which fluctuates slightly and becomes more negative at the end of the case study. The difference between the two can be explained with the help of the answers provided on who should address the software energy consumption in the awareness survey. Half of the RetailSystem participants point at solely 'software architects' with the addition of 'hardware manufacturers' and not by themselves. This raises the question whether we picked the right stakeholders to present our energy dashboard?

8.4 Willingness to address the SEC

The development team of DocGen is willing to address the energy consumption at the start of the case study. At the end they report to be less willing to address the energy consumption which is understandable as they don't know how. In contrast, the development team of RetailSystem were neutral at the start and negative at the end. The different sentiment between these two cases might be caused by the different industries and cultural aspects. Recall, DocGen is applied in the semi-governmental industry, while RetailSystem is applied in the retail industry. Besides the industry influence there is also the cultural differences between the cases. In The Netherlands there is a growing demand

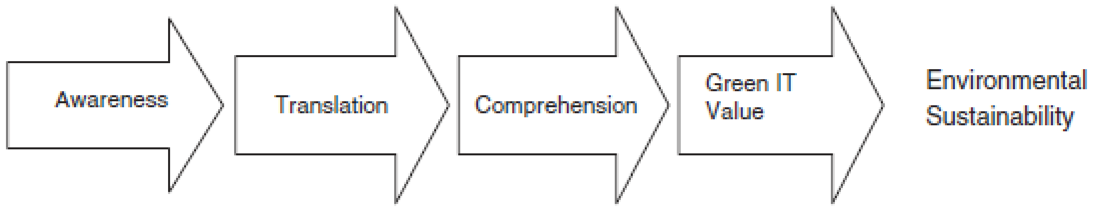


Figure 8.1: The Green IT value model derived from Chou and Chou (2012)

for sustainable services from industry while the product manager of RetailSystem did not recognize this aspect at the retail-industry in their respective countries.

8.5 Introducing Green Software to the Company

Both development teams are willing to address the SEC, if other teams address it too. This indicates that company-wide introduction of the SEC as a priority software quality would stimulate the adoption. Following the Green IT value model (8.1) by Chou and Chou (2012), we describe the four phases as applied in the model to introduce 'green' concepts in an organization.

The first phase is to raise awareness about the potential of the green strategy. The potential of the green strategy needs to be recognized by business in order to introduce it company-wide (Chou & Chou, 2012). With the execution of this research we've set the first preliminary steps on creating awareness on the potential of green software. However, the awareness must spread among the whole company in order to become an effective green strategy.

The second phase is to translate the green strategy of addressing the software energy consumption into concrete green initiatives. Based on the previous points in the discussion we propose four green initiatives to be executed.

The first green initiative is to identify all of the stakeholders within the organization. Our current dashboard targets development, but other stakeholders such as the company board or other strategic management require other information such as savings over time. These green initiative could be addressed by product managers. The product manager would be the ideal stakeholder as they communicate with internal and external stakeholders (Van De Weerd, Brinkkemper, Nieuwenhuis, Versendaal, & Bijlsma, 2006) and therefore reach a variety of different audiences to spread the word . Product managers also know the functionality of their software product and are able reach the right stakeholders to implement the optimization to reduce the energy consumption.

The second green initiative is the integration of the energy dashboard in the agile process. With the execution of our case study we've integrated the energy dashboard with every sprint review meeting, as it allows the stakeholders to look back at the efforts of the last sprint. The energy dashboard proved to be helpful as it was able to present the results of fixing a long-term bug with the RetailSystem case. Therefore, it has the potential to identify performance issues which could be addressed in the upcoming sprint(s). Two requirements need to be taken into account to support the integration of the energy dashboard with the development process First, the frequency needs to be adjusted to the need of the stakeholders to avoid frustration. Second, the energy and hardware utilization measurements need to be automated in order to be applicable in practice. In our current setup we need atleast one week to test one release of RetailSystem. In conclusion, the energy dashboard has the potential to identify performance issues which can be addressed during the sprint review meetings. To address performance issues, the results should be included in the sprint planning sessions. However, this should only be done in cases of performance issues and by request of the development team to avoid annoyance among the stakeholders.

The third green initiative is the creation of a knowledge bank, to address the knowledge gap of the developers.

The fourth green initiative is to go public, as in 'public within the company', with the energy dashboard of every software product. Currently the organization uses SonarCube to address software quality of different software products. The energy dashboard could be included in this project.

The combination of these four green initiatives would improve the position of green software within the company, as it becomes a tangible concept.

The third phase of the Green IT value model is the comprehension of the green strategy with monitoring the results of the energy dashboard and compare it with the defined goals.

The fourth phase is the generated value of green software with the organization receiving the benefits of the created awareness, translation and comprehension. This could be fulfilled sustainability goals such as commitment to the Corporate Social Responsibility treaty. In addition, it can also provide benefits to the customers such as an increased customer satisfaction or an increase in sales due to the distinctive characteristic of green software.

8.6 Limitations & Threats to Validity

The threats to validity can be related to two aspects of our case study; the measurement of the SEC and the survey. The next subsections discuss the construct validity, internal validity, external validity and reliability of both aspects.

8.6.1 Construct Validity

The construct validity addresses the degree to which the measures capture the concepts of interest and applies to the construction of the surveys.

The goal of survey A+B was to capture and quantify the awareness of participants. However, to the best beliefs of the authors, there are no standardized measurement tools or even a clear definition from the field of psychology on the concept of awareness. By selecting a **general definition** from the Oxford Dictionary we were able to apply it in our context. The model of changing environmental detrimental habits aligns with the definition and provided several concrete dimensions which fitted within the context of awareness on environmental issues. These two independent sources combined were the basis of our survey. From this basis we translated the concepts to our situation by following a structured approach for all of the constructs and created a chain of evidence from constructs to statements. Replication of this process might present different results. Therefore we do not position the statements as a general set of items to capture the awareness on software energy consumption. A thorough validation process is required to fully validate all of the constructs and statements. This process should be qualitatively assessed by academia from multiple disciplines and experts from the professional field and quantitatively assessed among a large population. The construction of survey C followed the same approach as survey A+B but was based on the UTAUT-model defined by Venkatesh et al. (2003). The constructs and statements from the model were selected and discussed by the authors and selected by their relevance. By removing several items we adjusted the internal consistency of the constructs as determined by Venkatesh et al..

8.6.2 Internal Validity

The internal validity addresses uncontrolled factors that might affect the results.

The internal validity regarding the survey might be affected by the **Hawthorne-effect**, with participants conducting modified behavior. Each participant filled in a number to link their responses to an individual participant and keep track of its progress.

With this approach we weakened the anonymity of the participant who might feel endangered by providing its honest opinion. To counter this effect we kept a separate list of the numbers and participants and assured the participants anonymity by not sharing this data. Another unaccounted factor is the **Principal-Agent (PA) problem** as the developers are not allowed to choose their own technology in order to reduce the SEC due to development guidelines of the SPO. The results indicate indeed a bias based on the PA-problem as developers have become more negative towards the relevance of creating energy efficient software and find it hard to identify improvements to reduce the SEC. To counter this effect we also included their personal opinion in our survey to identify their willingness regardless of the applied techniques, languages or architecture.

The internal validity of the SEC measurements were already identified with the previous work of E. A. Jagroep, van der Werf, Brinkkemper, et al. (2016). One of the major reported threats are the **reported values of Joulemeter**. Literature (E. Jagroep, van der Werf, Jansen, Ferreira, & Visser, 2015) indicates a gap in values between software based energy meters and hardware based energy meters. We applied best of both worlds to calibrate Joulemeter with the help of a physical energy meter but accurate measurements are hard to obtain. The second threat is the **measurement interval**. Both the hardware and software measurement approaches have a one second measurement interval. Fluctuations of a lower frequency interval of the electrical power aren't detected which results in a underestimation of the energy consumption. This threat was mitigated by executing long lasting repeated tests for both software products. The third identified thread are the effects of the **operating system**. The close-sourced nature of the Windows operating system makes it hard to control its behavior. It is possible to disable options for updates but it is impossible to get full control over all of the background processes. The RetailSystem case also had an additional middleware layer with Java. The influence of this extra layer is unknown and is out of scope due to the size. The fourth threat is the **energy consumption overhead** while calculating the SEC. In the original measurement approach, the SEC was calculated by the total power consumption of the system, minus the base power consumption. The result is the SEC during the execution of the investigated software. In this research we applied a different approach and calculated the energy consumption of each individual process. With this approach we focus on the processes which are created by the participants themselves and leave out the additional layers which cause the overhead.

8.6.3 External validity

The external validity address the extent to which the results can be generalized beyond the case study. The external validity is jeopardized by three aspects. The first aspect is the set of rules defined as our **inclusion criteria**. The inclusion criteria limit the number of potential companies and software product. However, the widespread implementation of agile practices within SPO's reduces the impact of this threat. On a side note, both software products are widely used on daily base in practice and therefore qualify as representative cases. The second aspect is related to the **company culture**. Both cases from the multiple embedded case study were conducted among the same SPO which limits the generalizability of the outcome due to company culture. To limit the impact of the company culture we selected two software product which were unrelated to each other. The participants of DocGen and RetailSystem were not only separated by the organizational structure but geographically among different countries as well. The latter introduces the third aspect as the origin of country introduces different **cultural backgrounds** among the participants. Due to the limited number of participants we were not able to investigate this aspect.

8.6.4 Reliability

The reliability address the extent to which the research is dependent on the specific researchers. To increase the reliability we have presented our case study protocol and described the deviations of each case. Results and interpretations can be traced back to the original data.

Chapter 9

Conclusion

The benefits of green software have been acknowledged by academia but are still unknown to companies. With this multiple embedded case study we have introduced the concept of green software, by means of the energy consumption of software products, at two software products at one SPO. With the introduction of the energy dashboard we have tried to create awareness among developers of the software product. The awareness was quantified with a score and measured before, during and after the introduction of an energy dashboard. With these results we are able to answer our sub-research questions (SQ) and ultimately our main research question (RQ).

9.1 Measuring Awareness

The first sub-research question targets the creation of awareness in relation to the energy consumption of software and is defined as:

SQ1: How can we measure awareness on software energy consumption?

Measuring awareness is difficult as we try to quantify something experienced by others through our own perception and therefore cannot be without bias. An alternative is to measure how humans perceive something based on a unified scale. Dunlap and Van Liere applied this method and developed a successful survey measuring awareness on sustainability. However, the topic of sustainability is too broad and therefore the test itself was inapplicable. The model of changing environmental detrimental habits as proposed by Matthies (2005) describe the stages and different aspects of norm activation and behavior. By translating these constructs to the topic of green software and developing a survey with statements related to the constructs, we were able to combine the two approaches and create an awareness survey targeting green software. A filled-in survey results in construct scores and an overall awareness score. By applying the survey multiple times among the same participants creates a set of different scores over time. A change in score indicates a change in either norm activation (awareness) or motivation (behavior). Thus, a change in scores indicates a change in awareness.

9.2 Triggering Awareness

The second sub-research question targets the stakeholders of the software product and is defined as:

SQ2: How can we trigger awareness on software energy consumption among stakeholders of a software product?

Energy consumption has been investigated for more than forty years among households. In the past few years, this field had some major innovations by means of smart energy meters. With smart energy meters it is possible to present real-time information about the energy consumption among households which increased the awareness of the residents on energy consumption. The field of eco-visualization encompasses the 'real time consumption statistics of key environmental resources for the goal of promoting ecological literacy' and propose dashboards as key instruments to visualize energy usage. Therefore, we developed an energy dashboard to present the hardware utilization and energy consumption of a software product. At the same time we developed the Resource Utilization Score (RUS), to quantify the resource utilization and use it for comparison among the different releases of the product software. The energy dashboard consists of the radar chart for the artistic visualization to grab the attention of the audience together with a details presented in a table to fulfill the pragmatic visualization and provide clear understanding. With the development of the radar chart and table we follow the 7th strategy of eco-visualizations: Raising public awareness and facilitate discussion.

9.3 Integrating the Stimulus

Th third sub-research question targets the integration of the stimulus to trigger awareness in the software development process. The sub-research questions is:

SQ3: How to integrate the stimulus in a software development environment?

During the execution of the case study we made the stakeholders aware of the software energy consumption by presenting the energy dashboard at the sprint review meetings. We chose the sprint review meetings as it allows the stakeholders to look back on their efforts of the sprint and demonstrate their deliverables. Integrating the energy dashboard in the sprint review meeting was a successful approach as the energy dashboard presented results related to the delivered work. For example, a long-term bug at RetailSystem was solved and the result on the energy dashboard confirmed this with a strong CPU reduction. However, using the energy dashboard to look back at the results won't be enough to maintain the awareness. In order to use the energy dashboard to its full potential and position the software energy consumption as a green initiative within the organization, it needs to be included in the sprint planning as well. However, measurements need to be automated in order to be usable, as not every sprint delivery results in a strong increase in resource consumption. Another requirement for integrating the stimulus is the creation of a knowledge bank with clear examples of addressing the energy consumption. However, recognition by the business is required in order to position the software energy consumption as a green strategy and maintain the awareness.

9.4 Concluding the Research

To conclude this research we answer our main research question:

RQ: How can we create awareness on the energy consumption of product software among stakeholders during product software development?

We created awareness on the the energy consumption of product software among the development teams by introducing an energy dashboard at the sprint review meetings. The energy dashboard presented several performance metrics in combination with the energy consumption of the software product. Concurrently we presented the Resource Utilization Score to quantify the resource utilization and to compare the scores among the different releases of the software product during the sprint review meetings. An awareness survey was created to quantify awareness as a score. We applied this survey at the start, during and at the end of the four sprint reviews. The survey was based on the model of changing environmental detrimental habits proposed by Matthies (2005) and transformed to the concept of green software. Besides the awareness survey, we introduced an user acceptance survey inspired by the UTAUT-model (Venkatesh et al., 2003) to measure the user acceptance of our energy dashboard. We defined awareness on green software as: 'understanding of the subject of software energy consumption at the present time based on information or experience'. The results of the awareness survey show a change in scores indicating a change in the understanding of the subject of the software energy consumption at the present time based on the presented information on the energy dashboard. Thus, we conclude that we created awareness on the software energy consumption of product software among stakeholders during product software development.

9.5 Future Outlook

With this research we've set the first steps of creating awareness on the topic of green software. The development teams of product software recognize the software energy consumption, although they do not always provide it the credits, it should deserve. If the concept of green software would become more concrete for developers, the awareness and acceptance would increase. In order to so we need a knowledge bank with practical examples of optimization of code to reduce the hardware utilization and energy consumption. At the same time, we need an automated approach to measure the software energy consumption of a software product. This would make the software energy consumption more tangible and stimulate the adoption of green software.

Some of the theories of eco-visualizations suggested the use of game-elements as an addition to create awareness, but we did not explore these possibilities. The focus of our research was on measuring and triggering awareness and not on how gamification could support these goals. Including gamification would deviate to much from our intended goal and therefore left out of this study. Exploring the possibilities of gamification to create awareness on green software is a possible research topic for the future.

References

Becker, C., Chitchyan, R., Duboc, L., Easterbrook, S., Penzenstadler, B., Seyff, N., & Venters, C. C. (2015). Sustainability design and software: The karlskrona manifesto. In *Proceedings of the 37th international conference on software engineering-volume 2* (pp. 467–476).

Bekkers, W., van de Weerd, I., Brinkkemper, S., & Mahieu, A. (2008). The influence of situational factors in software product management: an empirical study. In *Software product management, 2008. iwspm'08. second international workshop on* (pp. 41–48).

Boone, H. N., & Boone, D. A. (2012). Analyzing likert data. *Journal of extension*, *50*(2), 1–5.

Bozzelli, P., Gu, Q., & Lago, P. (2013). A systematic literature review on green software metrics. *VU University, Amsterdam.*

Brundtland, G., Khalid, M., Agnelli, S., Al-Athel, S., Chidzero, B., Fadika, L., ... others (1987). Our common future (\'brundtland report\').

Chou, D. C., & Chou, A. Y. (2012). Awareness of green it and its value model. *Computer Standards & Interfaces*, *34*(5), 447–451.

Commons, W. (2011). *Moore's law.* Retrieved 13-01-17, from `https://upload.wikimedia.org/wikipedia/commons/`

Deterding, S. (2014). Eudaimonic design, or: Six invitations to rethink gamification.

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: defining gamification. In *Proceedings of the 15th international academic mindtrek conference: Envisioning future media environments* (pp. 9–15).

Dick, M., & Naumann, S. (2010). Enhancing software engineering processes towards sustainable software product design. In *Enviroinfo* (pp. 706–715).

Dunlap, R. E., & Van Liere, K. D. (1978). The "new environmental paradigm". *The journal of environmental education*, *9*(4), 10–19.

Dunlap, R. E., Van Liere, K. D., Mertig, A. G., & Jones, R. E. (2000). New trends in measuring environmental attitudes: measuring endorsement of the new ecological paradigm: a revised nep scale. *Journal of social issues*, *56*(3), 425–442.

Goodland, R. (2002). *Encyclopedia of global environmental change, chapter sustainability: Human, social, economic and environmental.* Wiley & Sons.

Graus, W., & Worrell, E. (2008). The principal–agent problem and transport energy use: Case study of company lease cars in the netherlands. *Energy Policy*, *36*(10), 3745–3753.

Hazas, M., Friday, A., & Scott, J. (2011). Look back before leaping forward: Four decades of domestic energy inquiry. *IEEE pervasive Computing*, *10*, 13–19.

Hindle, A. (2012). Green mining: investigating power consumption across versions. In *2012 34th international conference on software engineering (icse)* (pp. 1301–1304).

Holmes, T. G. (2007). Eco-visualization: combining art and technology to reduce energy consumption. In *Proceedings of the 6th acm sigchi conference on creativity & cognition* (pp. 153–162).

Jagroep, E., van der Werf, J. M., Brinkkemper, S., Blom, L., & van Vliet, R. (2016). Extending software architecture views with an energy consumption perspective. *Computing*, 1–21.

Jagroep, E., van der Werf, J. M. E., Jansen, S., Ferreira, M., & Visser, J. (2015). Profiling energy profilers. In *Proceedings of the 30th annual acm symposium on applied computing* (pp. 2198–2203).

Jagroep, E. A., van der Werf, J. M., Brinkkemper, S., Procaccianti, G., Lago, P., Blom, L., & van Vliet, R. (2016). Software energy profiling: Comparing releases of a software product. In *Proceedings of the 38th international conference on software engineering companion* (pp. 523–532). New York, NY, USA: ACM. doi: 10.1145/2889160.2889216

Jagroep, E. A., van der Werf, J. M., Broekman, J., Brinkkemper, S., Blom, L., & van Vliet, R. (2016). A resource utilization score for software energy consumption. In *Proceedings of the 4th international conference ict for sustainability.* Atlantis Press. doi: 10.2991/ict4s-16.2016.3

Jagroep, E. A., van der Werf, J. M. E., Spauwen, R., Blom, L., van Vliet, R., & Brinkkemper, S. (2015). An energy consumption perspective on software architecture. In *Software architecture* (pp. 239–247). Springer.

Jevons, W. S. (1906). *The coal question: an inquiry concerning the progress of the nation, and the probable exhaustion of our coal-mines.* Macmillan.

Kansal, A., Zhao, F., Liu, J., Kothari, N., & Bhattacharya, A. A. (2010). Virtual machine power metering and provisioning. In *Proceedings of the 1st acm symposium on cloud computing* (pp. 39–50).

Kazandjieva, M., Heller, B., Gnawali, O., Hofer, W., & Kozyrakis, P. (2011). Software or hardware: The future of green enterprise computing. *Computer Science Technical Report CSTR*, *2*.

Koomey, J. (2009). *Computations per kwh over time.* Retrieved 13-01-17, from `http://www.koomey.com/post/14466436072`

Koomey, J., Berard, S., Sanchez, M., & Wong, H. (2011). Implications of historical trends in the electrical efficiency of computing. *IEEE Annals of the History of Computing*, *33*(3), 46–54.

Lago, P., & Jansen, T. (2010). Creating environmental awareness in service oriented software engineering. In *Service-oriented computing* (pp. 181–186). Springer.

Lago, P., Kazman, R., Meyer, N., Morisio, M., Müller, H. A., & Paulisch, F. (2013). Exploring initial challenges for green software engineering: summary of the first greens workshop, at icse 2012. *ACM SIGSOFT Software Engineering Notes*, *38*(1), 31–33.

Lago, P., Koçak, S. A., Crnkovic, I., & Penzenstadler, B. (2015). Framing sustainability as a property of software quality. *Communications of the ACM*, *58*(10), 70–78.

Maloney, M. P., & Ward, M. P. (1973). Ecology: Let's hear from the people: An objective scale for the measurement of ecological attitudes and knowledge. *American psychologist*, *28*(7), 583.

Manotas, I., Bird, C., Zhang, R., Shepherd, D., Jaspan, C., Sadowski, C., ... Clause, J. (2016). An empirical study of practitioners' perspectives on green software engineering. In *Proceedings of the 38th international conference on software engineering* (pp. 237–248).

Matthies, E. (2005). How can psychologists better put across their knowledge to practitioners? suggesting a new, integrative influence model of pro-environmental everyday behaviour. *Umweltpsychologie*, *9*(1), 62-81.

Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, *38*(8).

Mosley, H., & Mayer, A. (1998). *Benchmarking national labour market performance: a radar chart approach.* Wissenschaftszentrum Berlin für Sozialforschung.

Murugesan, S. (2008). Harnessing green it: Principles and practices. *IT professional*, *10*(1), 24–33.

Nagel, T. (1974). What is it like to be a bat? *The philosophical review*, *83*(4), 435–450.

Newton, T. (2010). *Perfmon: Identifying processes by pid instead of instance.* Retrieved 2017-01-13, from `https://blogs.technet.microsoft.com/askperf/2010/03/29/perfmon-identifying-processes-by-pid-instead-of-instance/`

Noureddine, A., Rouvoy, R., & Seinturier, L. (2015). Monitoring energy hotspots in software. *Automated Software Engineering*, *22*(3), 291–332.

Özden, M. (2008). Environmental awareness and attitudes of student teachers: An empirical research. *International Research in Geographical and Environmental Education*, *17*(1), 40–55.

Pang, C., Hindle, A., Adams, B., & Hassan, A. E. (2015). What do programmers know about the energy consumption of software? *PeerJ PrePrints*, *3*, e1094.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, *24*(3), 45–77.

Penzenstadler, B., & Femmer, H. (2013). A generic model for sustainability with process- and product-specific instances. In *Proceedings of the 2013 workshop on green in/by software engineering* (pp. 3–8).

Pierce, J., Odom, W., & Blevis, E. (2008). Energy aware dwelling: a critical survey of interaction design for eco-visualizations. In *Proceedings of the 20th australasian conference on computer-human interaction: Designing for habitus and habitat* (pp. 1–8).

Porter, M., & Van der Linde, C. (1996). Green and competitive: ending the stalemate.

Business and the Environment, 61–77.

Preist, C., & Shabajee, P. (2010). Energy use in the media cloud: Behaviour change, or technofix? In *Cloud computing technology and science (cloudcom), 2010 ieee second international conference on* (pp. 581–586).

Rasmussen, K., Wilson, A., & Hindle, A. (2014). Green mining: energy consumption of advertisement blocking methods. In *Proceedings of the 3rd international workshop on green and sustainable software* (pp. 38–45).

Schütz, H., Speckesser, S., Schmid, G., et al. (1998). *Benchmarking labour market performance and labour market policies: theoretical foundations and applications.* Citeseer.

Seo, C., Malek, S., & Medvidovic, N. (2007). An energy consumption framework for distributed java-based systems. In *Proceedings of the twenty-second ieee/acm international conference on automated software engineering* (pp. 421–424).

Sutherland, N. S. (1996). *The international dictionary of psychology.* The Crossroad Publishing Co.

Van De Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L. (2006). Towards a reference framework for software product management. In *14th ieee international requirements engineering conference (re'06)* (pp. 319–322).

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. *MIS quarterly*, 425–478.

Vereecken, W., Van Heddeghem, W., Colle, D., Pickavet, M., & Demeester, P. (2010). Overall ict footprint and green communication technologies. In *4th international symposium on communications, control and signal processing (isccsp 2010).*

von Alan, R. H., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, *28*(1), 75–105.

Weigel, R., & Weigel, J. (1978). Environmental concern the development of a measure. *Environment and behavior*, *10*(1), 3–15.

Wirth, N. (1995). A plea for lean software. *Computer*, *28*(2), 64–68.

Wohlin, C. (2014). 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* (p. 38).

Xu, L., & Brinkkemper, S. (2007). Concepts of product software. *European Journal of Information Systems*, *16*(5), 531–541.

Yin, R. K. (2013). *Case study research: Design and methods.* Sage publications.

Appendix A

Appendix

A.1 Questionnaires

A.1.1 Questionnaire A

SD = Strongly Disagree, D = Disagree, N = Neutral, A = Agree, SA = Strongly Agree

Statement	SD	D	N	A	SA
I want to determine the energy consumption of our software development environment (e.g. by calculating the number of (test)servers, laptops and other resources and their energy consumption) .	○	○	○	○	[✓ ○]
I expect that the energy usage is only marginally influenced by software.	○	○	○	○	○
If I had more time to work on the code, I would be able to reduce the energy consumption.	○	○	○	○	○
Our non-functional requirements (performance) do not allow room for implementing initiatives to address the energy consumption.	○	○	○	○	○
Before this project I had never thought of the energy consumption of our software development environment.	○	○	○	○	○
I would like to know the energy consumption of our software product.	○	○	○	○	○
Energy consumption cannot be reduced due to the implemented techniques (programming language, architecture, design patterns, etc.)	○	○	○	○	○
Investigating the energy consumption of our development resources (laptops/desktops (test) servers) in our software development environment is of no interest to me.	○	○	○	○	○
I expect that software has a large influence on the energy usage.	○	○	○	○	○
Code optimization to lower the energy consumption would be impossible despite extra time.	○	○	○	○	○
It is possible to make a trade-off between our current non-functional requirements (e.g. performance) and the energy consumption of our software.	○	○	○	○	○
Before this project, I wondered multiple times about the energy consumption of our software development environment, but didn't do anything with it.	○	○	○	○	○
The energy consumption of our software product is of no interest to me.	○	○	○	○	○
The applied techniques (programming language, architecture, design patterns, etc.) to realize the software product, allow for the reduction of energy consumption.	○	○	○	○	○

The energy consumption should be addressed by: (multiple answer are allowed)

A) Developers
B) Software Architects
C) Developers and Software Architects
D) Others, namely….

	SD	D	N	A	SA
Addressing the energy consumption of our software should gain more attention.	○	○	○	○	○
There is no discussion on energy consumption of the software product in our team.	○	○	○	○	○
Reducing the energy consumption would be a benefit to the company and to the customer.	○	○	○	○	○
The costs of rewriting the code, to reduce the energy consumption of our software, are too high compared to the benefits.	○	○	○	○	○
Addressing the energy consumption would be a waste of time.	○	○	○	○	○
If other teams reduce the energy consumption of their software, I would attempt it too.	○	○	○	○	○
The backlog doesn’t need to contain non-functional requirements to improve the software.	○	○	○	○	○
The energy consumption of software is irrelevant.	○	○	○	○	○
The energy consumption of our software product is discussed during (in)formal meetings.	○	○	○	○	○
I believe, nor the company or the customer would benefit from the energy reduction of software.	○	○	○	○	○
The benefits of rewriting the code to reduce the energy exceed the costs.	○	○	○	○	○
I would like to reduce the energy consumption, if I am allowed to spend time on it.	○	○	○	○	○
Initiatives on energy reduction at other software products do not influence our actions on energy consumption.	○	○	○	○	○
Code optimizations to improve non-functional requirements should be acknowledged and included in our backlog	○	○	○	○	○

A.1.2 Questionnaire B

SD = Strongly Disagree, D = Disagree, N = Neutral, A = Agree, SA = Strongly Agree

Statement	SD	D	N	A	SA
I expect that the energy usage is only marginally influenced by software improvements of this sprint.	○	○	○	○	○
If I had more time, I would have been able to reduce the energy consumption during the last sprint.	○	○	○	○	○
During the last sprint our non-functional requirements (performance) did not allow room for implementing initiatives to reduce the energy consumption.	○	○	○	○	○
In the last sprint I looked at the energy consumption of our software product.	○	○	○	○	○
Energy consumption couldn't be reduced due to the implemented techniques (programming language, architecture, design patterns, etc.) in the last sprint.	○	○	○	○	○
After the efforts of this sprint I expected that software had a large influence on the energy usage.	○	○	○	○	○
Code optimization didn't contribute to a lower energy consumption in this sprint.	○	○	○	○	○
It was possible to make a trade-off between our current non-functional requirements (e.g. performance) and reducing the energy consumption of our software during the last sprint.	○	○	○	○	○
The energy consumption of our software product was of little concern to me in the last sprint.	○	○	○	○	○
In the last sprint, the applied techniques (programming language, architecture, design patterns, etc.) to realize the software product allowed the reduction of the energy consumption.	○	○	○	○	○

A.1.3 Questionnaire C

SD = Strongly Disagree, D = Disagree, N = Neutral, A = Agree, SA = Strongly Agree

Statement	SD	D	N	A	SA
I find the dashboard useful in my job	○	○	○	○	○
My interaction with the dashboard is clear and understandable	○	○	○	○	○
Using the dashboard is a good idea	○	○	○	○	○
People who influence my behavior think that I should use the dashboard	○	○	○	○	○
I intend to use the dashboard in next sprint	○	○	○	○	○
If I use the dashboard, I will increase my programming skills	○	○	○	○	○
I would find the dashboard easy to use	○	○	○	○	○
The dashboard makes work more interesting	○	○	○	○	○
People who are important to me think that I should use the dashboard	○	○	○	○	○
I predict I would use the dashboard in the next sprint	○	○	○	○	○
Working with the dashboard is fun	○	○	○	○	○
The senior management of this business has been helpful in the use of the dashboard	○	○	○	○	○
I plan to use the dashboard in the next sprint	○	○	○	○	○
I like working with the dashboard	○	○	○	○	○
In general, the organization has supported the use of the dashboard	○	○	○	○	○

A.2 Included Papers

A.2.1 A Resource Utilization Score for Software Energy Consumption

A Resource Utilization Score for Software Energy Consumption

Erik Jagroep, Jan Martijn E. M. van der Werf, Jordy Broekman, Sjaak Brinkkemper
Utrecht University,
Department of Information and Computing Sciences
Princetonplein 5, 3584 CC Utrecht, The Netherlands
Email: {e.a.jagroep, j.m.e.m.vanderwerf, j.broekman, s.brinkkemper}@uu.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 as the true consumer of power and its potential contribution to reach sustainability goals is increasingly being acknowledged. Studies so far have presented successful results and methods to address the energy consumption of the software, indicating that different stakeholders striving for green software have different information needs with respect to their goals. However, currently there is no uniform manner to communicate measurements to the different stakeholders such that key findings are clearly identifiable and easy to understand, which is likely to hamper green software practices. In this paper we propose a metric that expresses a score for the resource utilization, such as power consumption, of a software product. The metric is designed to be a single score and is flexible to encompass those aspects that a stakeholder considers relevant in the context of software energy consumption. The metric was applied on two applications and allowed for objective comparison of application configurations and versions. Also the behavior of these applications across different hardware configurations could be analyzed. In addition to the metric we investigate means to visualize measurements which enhances communication and helped with highlighting the key findings.**

***Index Terms*—Software energy consumption, Resource utilization, Visualization, Sustainability.**

I. Introduction

The recent focus on the Energy Consumption (EC) of software has had a positive impact on the spectrum of sustainable, i.e. energy efficient [1], solutions in the ICT sector. Although hardware consumes energy, software directs the hardware on using the available resources [2] and numerous studies become available that report improvements on energy related aspects with the software itself as the central topic [3], [4]. In a recent study, Hindle [5] presents a method to analyze EC across releases of software products, which provides a basis for sustainable endeavors a software producing organization [6] might undertake. Despite this, organizations still struggle with addressing the software products in terms of their EC [7].

Key in this struggle is that addressing the EC of software confronts a software producing organization, specifically software developers, with a multifaceted issue. Depending on its deployment, measuring the EC of software can be done using relatively cheap hardware devices. However, apart from EC measurements, the software is also characterized using performance measurements which allows for analysis of the software's energy consuming behavior and resource usage. Performance measurements in our case refer to hardware resource performance and provide insight in how the hardware components are stressed when processing instructions. Developers should be able to use this information to address this relatively unknown, non-functional aspect [8] of the software.

Based on previous work (i.e. [9], [10]), however, we found that these measurements are not easy to communicate to stakeholders. Our experience is that in some cases deep knowledge is required to understand the measurements, i.e. how should a specific (performance) measurement or metric be interpreted, and that the key findings that require further investigation are difficult to identify. Issues that are strengthened by developer knowledge that is lacking in this area [7]. As a result, we had developers and software architects searching for the right information and identified a potential inhibiting factor to start addressing the EC of the software.

In this paper we investigate a means to effectively communicate Software Energy Consumption (SEC) related measurements to stakeholders wanting to address the sustainability of their software. Effectively in our case means that the information is easy to understand, is reported uniformly to enhance recognition, and clearly communicates any key findings. Ideally we are able to express the results in a metric that allows to objectively compare the software across different contexts (e.g. releases, installations).

Based on the above we formulate our main research question as follows:

RQ: *How can we effectively express the resource utilization for executing a software product in relation to the SEC?*

In the RQ, we refer to various resources as it is clear that energy is not the only involved resource. However, as this differs per study, a possible metric should be flexible to encompass those resources that are considered important in a given context. Translating the focus on resource usage to a metric, we contribute by providing a Resource Utilization Score (RUS) for the SEC. The RUS helps in the analysis of SEC related measurements and their visualization.

The remainder of this paper is structured as follows. We first present the related work (Sect. II) and continue with

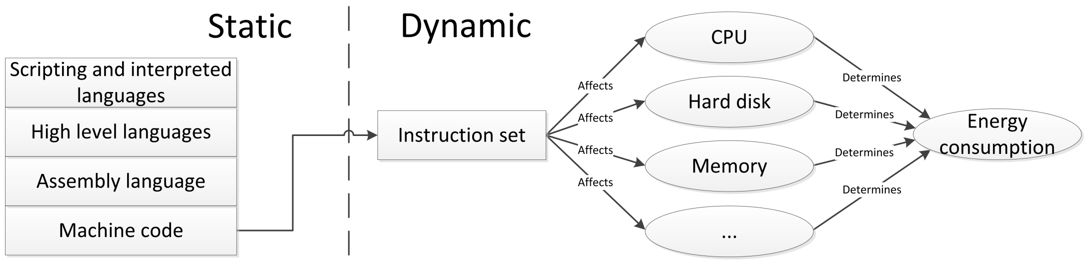


Figure 1. A translation from the language abstraction stack (static) to the energy consumers (dynamic).

the creation of the RUS (Sect. III). After constructing the score we apply the RUS in an experiment (Sect. IV) and evaluate the results (Sect. V). Finally, we discuss our findings (Sect. VI) and provide a conclusion including directions for future research (Sect. VII).

II. Related Work

The term sustainability in the ICT domain is aimed at controlling ecological, economical and social dimensions (of ICT) to the extent that future stakeholders are not compromised in the ability to meet their needs [11]. Sustainable ICT, also coined 'Green IT' [1], helps to improve energy efficiency, lower greenhouse gas emissions and promotes reuse and recycling. Recently a technical dimension has been added for software intensive systems [12], addressing the aspects of the constantly changing environment in which software is executed. Our focus on optimizing the EC of software, i.e. 'green software' [13], is mainly concerned with the ecological dimension, however economic, social and technical goals could also be addressed using a RUS.

A. Addressing Software Sustainability

A popular approach towards creating green software is to consider sustainability [13], or energy efficiency [14], [15], as a quality aspect for the software. Doing so allows software producing organizations to consider sustainability aspects during the design of the software, i.e. with its software architecture, and make trade-offs with other quality aspects (e.g [16]). However, analogous to the other quality attributes [17], working on software quality could require significant investments in terms of time, specialized knowledge to address specific issues and analysis across architectural views [18]. An EC perspective [9] could help guide any efforts in this regard, but the complexity of the matter could still pose difficulties.

If we look at the language abstraction stack (Fig. 1), we can explain the complexity. After a blueprint for the software is made in the form of its software architecture [8], the actual software development can take place. Through several layers of abstraction an instruction set is acquired that is executed by the hardware. This brings us from the static to the dynamic aspect of software. Performing the sequences of instructions affects the hardware components and available (virtualized) resources, which in turn determines the EC induced by the software. Hence, as developers have limited control over the instruction set, they can only await what effect changes in the source code might have. Some tools are available though, e.g. Big-O notation [19], but again complexity issues rise due to the large software systems that organizations produce.

To exert control, apart from EC measurements, studies in the area of green software report a variety of metrics depending on the context in which a study was performed and the stakeholders that are involved. For example, performance [9], [20] and software [5] metrics are used to characterize a software product, which is typically input for developers and architects. These two stakeholders could also benefit from knowing the EC on process level [4]. As the developer is responsible for writing the code, these insights could stimulate to, for example, minimize the number of invocations for a specific, high energy consuming method.

On the other hand we find metrics on infrastructure and organizational level, that are useful for higher level sustainability goals (e.g. by product management [21]). Green performance indicators [22] can, among others, be used to monitor infrastructure facilities and are of interest when EC needs to be considered on datacenter level. In their work, Lundfall et al. [23] present a tool to make the economic impact of green practices explicit with the purpose of justifying green practices on management level. The relation with green software is apparent though as the figures often still stem from low level computing and application measurements.

B. Resource Utilization

Attributing the EC to the software itself requires monitoring the usage of the available hardware resource; i.e. performance measurements. Performance measurements provide insight in how the hardware components are stressed when processing instructions and specific performance metrics can be identified for each individual component [9]. For example, [24] monitors the overall system throughput, CPU, memory and hard disk through the 'number of instructions', 'CPU utilization', 'memory utilization', and 'disk transactions per second' performance metrics. A different approach is to assume theoretical EC figures, e.g. based on the specifications provided by the manufacturer [25], however this approach fails to account for the dynamic behavior of the software.

Important in this field of research is to select the relevant hardware components to monitor and the right instructions to process. Traditionally the CPU has been identified as the most decisive component for the EC by a system [26], [27]. However, CPU based energy models do not capture all the power drawn by a system [28]. In [29] the contribution of each laptop component to the energy consumed is identified, e.g. the optical drive and LCD-backlight, and shows only 20% can be attributed to the CPU. On the other hand, in large-scale infrastructures the EC of network equipment is argued not to fluctuate heavily with increased traffic [30]. With regard to the instructions to process, a workload model should be made to reflect realistic conditions [24].

Resource monitoring is relevant on different levels related to green software. While investigating the EC of a server, a static and dynamic component can be identified [28]; static is the EC while the system is idle, i.e. minimum resources are used, and the dynamic EC fluctuates with the usage of the resources. As the static component is a large part of the EC, minimizing the absolute number of physical servers could significantly contribute to achieving the desired EC savings. In a cloud architecture the load of resources can be monitored (CPU, disk storage and network interface) and nodes switched on or off to minimize the overall power consumption [27]. This potential to scale up or down, based on performance monitoring, could help in achieving cost-effective scalability [31]. The dynamic part is relevant for green software practices and is determined by the resource utilization of the software.

C. Labeling Software Products

In [32] work has been done towards creating eco-labels for software in terms of a definition, criteria, form of representation, target groups and stakeholders. Sustainability is considered in the broadest sense of the word and, for example, also includes the sustainability aspects of the development process for the software. Following the main criteria that are identified, Kern et al. [32] continue with selecting those criteria that should be considered based on the life cycle phase of the software. This selection appears to be in line with the metrics and information needs as discussed above.

We deviate from [32] with respect to the form of representation. The authors build on international examples of eco-labels, although valuable in their own rights, which often do not allow for many details (i.e. low-level metrics) and are solely focused on specific aspects (e.g. CO_2 emissions). Consequently, apart from providing a starting point, the suggested eco-labels would be of limited practical value for those wanting to address the sustainability of their software.

Having said this, we consider the RUS and the eco-labels complementary in the area of green software. Eco-labels could help in selecting the tools, frameworks and services that positively impact the EC of a software product. For example as a criterion for the service-adaptation tactic [33] in a cloud context. The RUS, on the other hand, could serve as a more hands-on tool for software developers and architects.

III. Resource Utilization Score

In the search to determine a RUS for software EC, we investigate a means to combine performance metrics into a single score. However, we also acknowledge the importance of clearly communicating any key findings and the importance of lowering the threshold to interpreting the measurements through its presentation. To this end, we include a means to visualize measurements in our investigation.

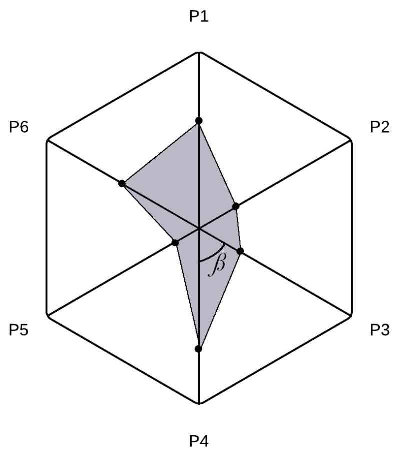


Figure 2. An example of a radar chart with example profile.

A. Visualizing Measurements

In general visualizing a measurement, e.g. per software element [4], simplifies its communication and interpretation compared to raw measurements. However, visualizing measurements individually limits the user in combining metrics and neglects any relation between them. In the case of SEC, the hardware components receive instructions from some instruction set translated from the software. As such there is bound to be a relation between the instructions that the components have to process. Consequently, we aim for a visualization method that is able to encompass all measurements in one figure and can serve as a basis for determining a score.

For our purposes we found a solution in the radar chart. According to Schutz, Speckesser and Schmid [34] the radar chart serves four goals:

1) Visualize interrelated performance measures through standardized scales.
2) Produce an effective description of selected performance dimensions in one synthetic indicator.
3) Analyze the change in overall performance between two points in time by comparing the surface of the same object.
4) Compare different objects through the shape of the surface for these objects.

Translating these goals to our context we can use a radar chart to visualize the performance dimensions related to the SEC, combine the dimensions into one single indicator (i.e. a score), analyze changes on different points in time (e.g. across releases [10]) and compare software products to one another. Under the condition that the same metrics are used for the chart. An example of the radar chart is provided in Fig. 2, showing the dimensions (P1 through P6) for an unspecified object and the forthcoming surface (grey area) resulting from the scores on these dimensions.

In [35] the idea of the radar chart is applied to benchmark the performance of national labor markets. Although the results look promising, limitations of using the surface of the radar chart are also identified:

- The right dimensions should be selected for benchmarking a specific aspect.
- The right metrics should be selected to characterize these dimensions.
- The dimensions could contribute differently to an indicator (score) and as such could require a weighted inclusion.

The first two limitations concern selecting the right dimensions, i.e. axes, and the right performance metrics to characterize these axes. We address these limitations for our research in Sect. III-B. The third limitation affects the calculation of the RUS and is addressed in Sect. III-D.

B. Determining the Axes

It should be clear that the aspect under investigation in our research is the SEC. The first step is to determine the dimensions, i.e. the axes, that will form the radar chart for this aspect. From the related work we are able to distill the following dimensions that are directly or indirectly affected by the instruction set:

- CPU
- Memory
- Hard disk
- Network
- Power consumption
- Execution time

Each dimension has its own performance metrics (e.g. % usage versus bytes total per second [10]) and each metric is expressed in its own unit and scale (e.g. utilization percentage versus number of (M)Bytes). Selecting the dimensions and corresponding metrics should be done for each individual case as this depends on the product under study. Note that the dimensions are not orthogonal, e.g. higher resource utilization results in increased power consumption.

Following the first goal presented for the radar chart, we should aim for a standardized means to present the measurements. Looking at the diversity of the list, one of the few options to determine a standardized score on each dimension is to use ranges. With regard to resource usage, a minimum resource usage can be determined in the situation where the hardware is idle and a maximum where the hardware is stressed to its maximum capacity [36]. The available resource, i.e. the margin between the minimum and maximum resource usage, forms the range. Note that the range should be determined individually for each performance metric. When an activity is performed using the software, the required resources can be divided by the range. This transforms measurements to a value between 'zero' and 'one' for that specific metric.

There is however a downside to working with ranges, as not every aspect can be expressed using a range. The execution time, for example, could have an infinite maximum, i.e. run as long as required without limitations. As such, we suggest to exclude these aspects from the chart itself and instead report these separately. For example, the units of work [15] to create a workload model are described separately to correctly interpret the measurements and the context in which they were found. We continue our work using the range method.

A final aspect is the order in which the axes are included in the radar chart. Using the same data in a different order can result in a difference of up to 300% [35], posing a threat to the third and fourth goal identified with the radar chart. We take this issue into account in the next section where we calculate a score for SEC.

C. Calculating the RUS

Continuing on the path of the radar chart, following goal three, we are able to obtain an objective performance measure by calculating the surface of the chart. This calculation is described by Mosley and Mayer [35] as the Surface Measure of Overall Performance (SMOP) and is calculated using Eq. 1.

$$\text{SMOP} = ((P_1 \cdot P_2) + \ldots + (P_n \cdot P_1)) \cdot \sin(\frac{\pi}{n}) \qquad (1)$$

In the equation, the P-values represent the axes of the radar chart. The resulting number represents the surface of the figure created by all of the connected dots on the chart, i.e. the grey surface in Fig. 2.

The problem with Eq. 1 is that the axes are ordered implicitly. As there is no clear order between the different measures the axes represented, ordering them differently results in different values for the surface. As we do not have an explicit, clear order for the measures, a solution is sought by calculating the average surface based on all possible surfaces. Rephrased, we consider all possible relations between the axes. Instead of calculating for each possible order the corresponding surface, we observe that each possible triangle of axes is taken into account an equal number of times. Hence, calculating the surface for all different triangle suffices. Translating this to an equation results in Eq. 2:

$$\text{SMOP} = \sin(\frac{\pi}{n})(\sum_{i=1}^{n}\sum_{j=1}^{n}(P_i \cdot P_j)) \qquad (2)$$

Observe that each score is multiplied by a constant factor. As we want to use the score for comparing different solutions, this constant can be left out. However, in its current form we see that axes could be paired with themselves, and that all pairs are counted twice (i.e. the symmetrical pairs $P_i \cdot P_j$ and $P_j \cdot P_i$). Taking these elements into account, the equation can be simplified as follows:

$$\text{RUS} = \sum_{i=1}^{n}\sum_{i<j}^{n}(P_i \cdot P_j) \qquad (3)$$

A side effect of excluding the constant factor is that we do not longer calculate the surface of the chart, but rather a *score* based on the relation between the axes. As a result we are able to include non-standardized dimensions (e.g. execution time) in the equation that are not included in the radar chart. We labeled the score as the Resource Utilization Score.

D. Weight Factor

With the axes for benchmarking SEC identified and the ability to calculate a score, one important limitation remains that should be addressed: weighting the contribution of the different indicators. In our case this limitation counts for the performance metrics, but extends to the level of dimensions (i.e. axes). Concerning the first, we can only argue that the right performance metrics should be used to determine the scores on the respective axes. A hard disk, for example, could be characterized using the 'Disk I/O per second' and the '# Mb per second' metrics [9].

With regard to the axes we acknowledge that some combinations could be considered more important than others and have a bigger influence on EC [37] depending on the context. In large scale infrastructures, for example, memory plays an important role. As such, specific combinations that include memory could be argued have a greater contribution to the RUS. As these combinations are context dependent, we introduce a weight to the different ax-combinations, which results in the following score:

$$RUS = \sum_{i=1}^{n} \sum_{i<j}^{n} W_{i,j}(P_i \cdot P_j) \tag{4}$$

This score (Eq. 4) provides us with the RUS to characterize the resource utilization in relation to SEC. The weight factor can be used to reduce or amplify the effect of certain combinations of measures. In-depth analysis of the performance data, for example through a regression model [10], can help in determining the weight factors.

IV. Experiment Design

To evaluate the RUS, an experiment was performed to select the most resource efficient algorithm to calculate the first N decimals of π from an application that provides many algorithms to calculate π. Additionally, as a more practical evaluation, we apply the RUS to an already available dataset [9]. In this section we describe the setup of the experiment for which we followed the guidelines provided in [38]–[40].

A. Experiment environment

For our experiment a test environment was prepared consisting of an application system, a logging system, and a measurement device (Fig. 3). The application system is the test hardware on which the software product is to be installed and as such the system to monitor. The logging system collects data from multiple sources and provides task instructions to the application system. Finally, the measurement device, a WattsUp? Pro (WUP)[1], is used to measure the power drawn by the application system and calculate the SEC. Since the

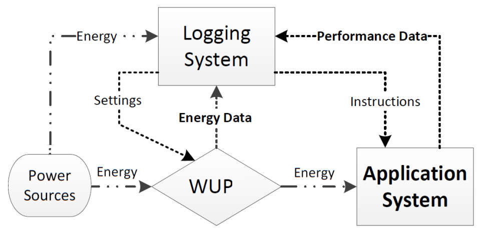


Figure 3. Experiment setup

WUP is a separate device, the energy usage of the application system was not influenced by its measurements.

In total three different application systems were included in the experiment, a laptop, desktop and a server, each representing a different computer class. For a system to be included in the experiment the device had to run the Microsoft Windows 7 *Professional* (or higher) operating system and be equipped with a multi-core Intel processor. These requirements provided us with systems that are capable of remote performance monitoring and are representative for modern systems in terms of computational capabilities. Details of the selected systems are shown in Table I.

Performance measurements were collected using the Windows Performance Monitor[2] (Perfmon). Perfmon enables remote performance monitoring of systems with a one second interval in between measurements and is freely available with the Windows operating system.

B. Test Application

To simulate activity, we used *Systester* (version 1.5.1)[3]; an application that calculates Pi decimals using the *The Quadratic Convergence of Borwein* and *Gauss-Legendre* algorithms. For the first algorithm both a single- and multi-core variant was available This enabled us to not only compare the difference between the two algorithms, but also between a single- and multi-core configuration. In order to have controllable runs the choice was made to calculate $8 \cdot 10^6$ Pi decimals per run.

For the actual experiment the remotely executable command line version of Systester was used, i.e. from a batch script using the logging system. In addition, a modified version of Systester was compiled where the application waits five seconds after initiating and before ending the process. The presence of this five second interval allowed Perfmon to collect all data related to a task and made it easier to identify the specific runs during processing, thereby directly contributing to the quality of the data and forthcoming analysis.

C. Metrics and Utilization Ranges

Visualizing measurements on a radar chart requires the metrics to be expressed on a standardized scale. To do so, we require the minimum (zero) and maximum (one) resource utilization figures which allows us to express measurements as a value on this continuum. The idle measurements (minimum)

[1]https://www.wattsupmeters.com/secure/products.php?pn=0&wai=0

[2]https://technet.microsoft.com/en-us/library/cc749154.aspx

[3]http://systester.sourceforge.net/

Table I
SPECIFICATIONS OF THE APPLICATION SYSTEMS.

	System		
Property	Laptop	Desktop	Server
Brand/model	ASUS F3JA	*Custom PC*	HP DL380 G5
Processor	Core 2 Duo T7200	Core 2 Duo E6750	Intel Xeon E5335
FSB / TDP	667 / 34	1333 / 65	1333 / 80
Chipset	Intel i945PM	Intel P35	Intel 5000P
Memory	2GB DDR2	2GB DDR2	4GB EDO
Operating System	Windows 7 Professional (32 bit) Servicepack 2	Windows 7 Professional (32 bit) Servicepack 2	Windows server 2008 (64 bit) Servicepack 1

were performed by leaving the application systems idle (without going to a sleep state) for at least 30 hours and monitoring the resource utilization and power consumption during this period. To determine the maximum resource utilization figures the application systems were stressed to their maximum capacities using HeavyLoad[4].

Based on the characteristics of Systester, the following metrics were selected to create a radar chart:

- **CPU**: '% CPU time'. The maximum utilization is 100% per core adding up to a percentage above 100% for multi-core systems. The range was determined with the '% CPU time' while idle and using HeavyLoad.
- **Memory**: 'Available bytes'. The number of bytes that is available of which the value decreases as processes require memory. The maximum is the available bytes while idle which also indicates the range for this metric.
- **Disk**: '% disk idle time'. The time that the disk was idle. The maximum utilization for the hard disk is 100% and its range is found by subtracting the '% idle time' of an idle system from this 100%. While the '% disk time' metric can also be used, this metric exaggerates[5] disk utilization.
- **Power consumption**: The 'power consumption' (in Watt) by the system while performing a run. The range was determined per system by measuring the 'power consumption' while idle and while using HeavyLoad.

Given the nature of the application we decided to exclude network metrics. Note that the actual SEC is calculated using the WUP measurements and stems from a different source than the performance measurements.

To calculate the RUS, the standardized metrics of the radar chart will be combined with the non-standardized 'execution time' metric. The 'execution time' is defined as the time required to perform a specific task and could be a determining aspect for SEC [9]. In our experiment the execution time is the time for Systester to calculate $8 \cdot 10^6$ Pi decimals.

D. Experiment protocol

To actually perform the experiment a protocol was followed containing every activity required to perform a series of runs. A run is one time for the application to calculate $8 \cdot 10^6$ Pi decimals plus the five seconds before and after performing this task. A series can be configured to include multiple runs. For each series a script was used, *PiBatch*, which automates monitoring with PerfMon, optionally includes rebooting the system, and performs a specified number of runs. The script minimizes human interference, provided that the following preparations are made:

- Install PsTools[6] for executing commands remotely.
- Install software to remotely manage the WUP.
- Remove the battery from the laptop to eliminate battery charging/discharging effects.
- Configure Windows power settings and disable unnecessary services (e.g. Windows Search and Update).
- Configure Perfmon data collector set.

Additionally, the effect of rebooting the application systems was investigated. In a small experiment we found that a system was 'unpredictable', i.e. random active processes, in the first 15 minutes after rebooting. The PiBatch script takes this into account by waiting at least 15 minutes before starting the first run of a series. This resulted in the following protocol:

- Clear WUP meter data and test connections.
- Configure and initiate PiBatch script .
- Collect data from PerfMon and WUP.

At the time of the experiment, rebooting was made optional in the script as rebooting the server appeared not possible with a virtual machine running. The virtual machine was running on one single, dedicated server and was isolated from other infrastructural facilities ensuring that the only hardware that is affected is the hardware being measured. To get a representative data set, we decided to continue until at least thirty clean measurements per combination were obtained.

E. Post-processing the Measurements

After performing a series of runs, post-processing was required before analyzing the data.

Determine run execution time; The runs appeared of variable length and hence we needed to determine the exact execution time for each run using the '%CPU Time' of the application process (provided by PerfMon). The execution interval started when the '%CPU Time' was more than zero and ended when it went back to zero again.

Synchronize WUP and Perfmon timestamps; Since the WUP and PerfMon data stemmed from separate sources, the

[4]http://www.jam-software.com/heavyload/

[5]https://technet.microsoft.com/en-us/library/cc938959.aspx

[6]https://technet.microsoft.com/en-us/sysinternals/bb896649.aspx/

Table II
THE RUS FOR EACH COMBINATION (LOWER IS BETTER).

	Laptop	Desktop	Server
Borwein, single-core	393.44	421.15	354.57
Borwein, multi-core	358.87	405.73	377.52
Gauss-Legendre	158.53	194.56	147.74

Table III
EC CONSUMPTION FIGURES FOR EACH COMBINATION IN JOULE.

	Laptop	Desktop	Server
Borwein, single-core	17,989	26,092	103,165
Borwein, multi-core	14,724	20,555	82,204
Gauss-Legendre	7,442	10,891	44,870

timestamps of the measurements required synchronization. The start of an interval in the WUP measurements, i.e. an increase in power drawn, was matched to the moment of initiation of the associated processes in the PerfMon data. Cross-checking on corresponding end points ensured that the synchronization was correct.

Assess quality of runs; Despite our efforts to control any effects that could influence the experiment, we observed activity unrelated to Systester during the experiment. First, we used the '% CPU time' on process level to check whether a system was solely processing tasks related to Systester during a run. In addition we monitored the EC for discrepancies as the performance measurements could not always explain increases in power consumption. Runs including showing odd patterns were excluded from further processing.

V. EVALUATING THE RUS

The results of the experiment are summarized in Fig. 4 which shows the radar charts for each combination of the three systems and Systester options. The charts show the average score on each metric for the specific configuration, e.g. the laptop on average used 69% of the available CPU resources with the multi-core quadratic convergence of Borwein. In total 32 runs were performed for each algorithm on the laptop, which were are all clean. On the desktop 34 runs were performed with the Gauss-Legendre algorithm and 32 runs with both Borwein algorithms, providing 31 clean runs per algorithm. The server appeared the most problematic system where we performed 80, 72 and 55 runs for the Gauss-Legendre, single-core and multi-core Borwein algorithms to obtain 42, 42 and 55 clean runs respectively. Given the instability, we decided to obtain at least 40 clean runs for the server.

Since the execution time was not suitable to express on an axis, we provided this information alongside the corresponding radar chart. At a glance we can conclude that the Gauss-Legendre algorithm is the fastest option of the three, and that the multi-core variant of the Borwein algorithm is faster than its single-core variant. Surprisingly we find that the server, despite its computational capacity, on average is at least 30 seconds slower compared to the other systems with the Gauss-Legendre algorithm. A trend that also shows with the other algorithms. We were not able to find the cause of this discrepancy, but we argue these figures could be typical for the server and associated hardware possibly in combination with the multi-core capabilities of Systester itself.

The impact of the execution time can be made clear using the actual SEC figures (Tbl. III) on the account of Systester. Although the metrics indicate a fairly similar resource utilization pattern across machines, in terms of absolute SEC we find that the server consumes more energy. The same holds for the desktop compared to the laptop; similar scores, higher SEC by the desktop in absolute terms. If we consider the SEC findings in light of the radar charts we solidify the argument on adding the execution time to the visualization.

Looking at the dimensions themselves we find that in this particular case there seems to be a relation between the scores on the CPU and power dimensions, i.e. an increase on the CPU dimension pairs with an increase on the power dimension. Also, as expected, the CPU scores are highest with the multi-core Borwein algorithm, but do not double in comparison to the single-core version. The difference between the Borwein measurements could be an indication of the potential for multi-core (i.e. multi-threaded) applications.

With regard to the memory and disk metrics, the laptop and server charts indicate minimal impact on the memory and disk dimensions. However, the radar charts clearly indicate a different situation for the disk utilization by the desktop. While we find the high disk utilization for the desktop peculiar, we cannot attribute this utilization to Systester as the other systems do no exhibit this behavior. Based on the information a further analysis can be performed on the desktop.

A. *RUS Scores*

The corresponding RUS for each combination is provided in Tbl. II. The RUS was calculated using the standardized scores of the performance metrics and the execution time in seconds (non-standardized). Hence, the scores are larger than 'one'. As there were no indications to prefer a specific dimension over the others we set the weight factor for all pairs of dimensions to 'one'. Important to notice is that a lower score means that a specific combination scores better; i.e. requires less resources.

Comparing the RUS with the EC figures (Tbl. III) we find that in general lower RUS scores are accompanied by lower EC figures. The only exception is with the single- and multi-core Borwein algorithms on the server. Looking at the radar charts we find that the multi-core variant shows higher utilization scores on the power and CPU dimension, an increase that is also visible on the other systems. In relation to the other systems we can only conclude that the difference in execution time is enough to offset the EC figures but not the RUS. From a practical perspective, the single-core variant could be preferred above the multi-core variant when trade-offs should be made (e.g. when resources are shared).

If we solely use the RUS scores to choose an algorithm and platform, the decision would be to run the Gauss-Legendre algorithm on the server. However, based on the EC we should actually prefer this algorithm on the laptop or, if we favor speed, on the desktop. This observation learns that the RUS

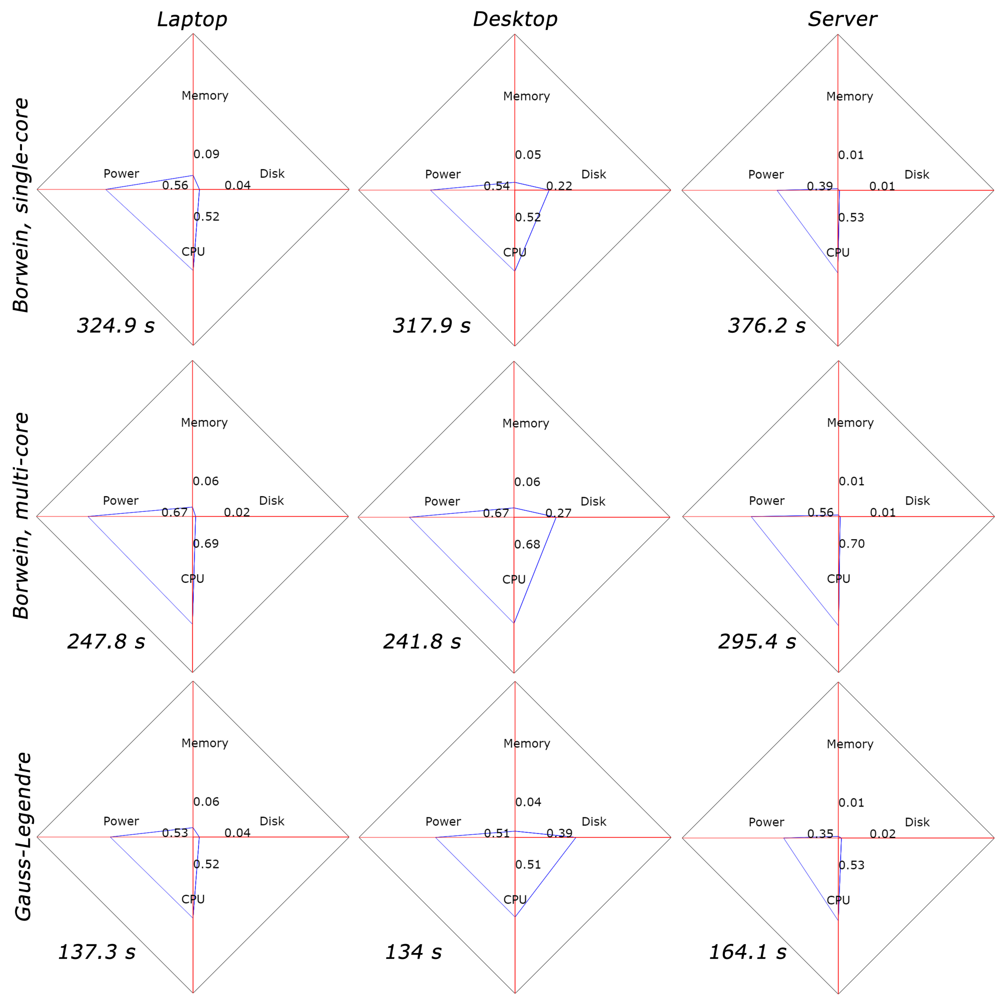


Figure 4. The collection of radar charts showing the resource utilization on each dimension and execution time for the nine investigated combinations.

should be considered complementary to the EC and that we cannot use the RUS to compare across classes of systems.

B. RUS for a Commercial Software Product

As an additional evaluation we applied the theory to a commercial software product (Document Generator) with the instructions to generate 5000 documents [9]. The metrics for the radar chart were '% CPU time' (CPU), 'available MBytes' (memory), '% disk idle time' (hard disk), 'power consumption' (in Watt) and the 'total Bytes per second' (network). Compared to Systester the network metric was included and and its range was determined using Lan Speed Test[7].

In this case an architectural change was applied to make Document Generator multi-threaded. The resulting decrease in CPU Utilization, i.e. from 49% to 19.2%, lowered the EC per document with 67.1% This finding is visible in the radar charts (Fig. 5) where a decrease in the the CPU and power utilization can be observed. A minimal increase in utilization of the disk and memory was found, whereas the network utilization remains unchanged. Especially the CPU utilization, or more specifically the division of the workload between CPU cores [9], seems decisive for the power dimension.

The RUS scores (Tbl. IV) were calculated using the execution time and appear to be in line with the EC measurements; i.e. a lower score means less SEC. This finding possibly suggests that the utilization patterns after adjusting the software are more 'natural' to the system.

VI. Discussion

In this research we investigated the possibility for a RUS to express resource utilization in relation to the SEC. The constructed score is based on those dimensions that are deemed relevant for the software and is flexible to be adjusted depending on the product and the environment in which it will be executed. Initial evaluation showed promising results to engage in green software practices. There are, however, several limitation to our work which we discuss below.

Hardware dependency; Like EC, the RUS is dependent on the hardware that the software is executed on. Although the measurements are standardized, the ranges themselves showed

[7]http://totusoft.com/lanspeed/

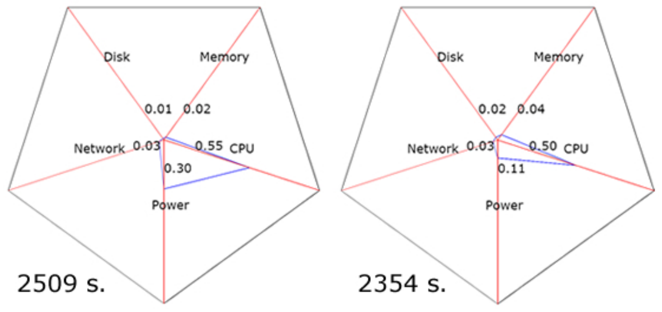


Figure 5. The radar charts for the Document Generator software product before (left) and after (right) making the software multi-threaded [9].

considerable differences across systems. Comparing the RUS and EC figures led to the insight that the RUS cannot be used to compare a software product across classes of systems. Additionally, it can be impossible to determine the ranges in environments with ‘unlimited’ resources (e.g. cloud data centers). Different benchmarks should be found when this is the case, for example benchmark release of software to one another to visualize the effect of software changes.

Visualization; The RUS is based on the theory related to the radar chart. Even though we investigated different theories, other options could exist that better fit the purpose. For example, theories that better consider the relation between dimensions or express a score based on the dimensions.

Robustness; The RUS was successfully applied to both a synthetic (Systester) and commercial (Document Generator) application, which shows the generic ability of the theory to be applied. Although we are confident in the validity of our results, more applications of RUS are required to prove or disprove the robustness of the RUS.

Weight factor; The weight factor for calculating the RUS is a topic that still requires further investigation. In our experiment we could not motivate a higher score on one combination of dimensions over the other and decided to set the weight factor to ‘1’ for each combination. However, other situations might require a thorough investigation to determine the correct weight factors.

A. *Experiment Limitations*

Despite our best efforts, there are limitations to the experiment as described:

Windows processes; Thirty minutes after rebooting we observed an increase in activity for an unspecified period of time. The cause is unknown, but we assume that Windows-related processes are triggered by a timed mechanism which we cannot control. However, we did not find significant differences between runs executed after twenty or 200 minutes and between Windows 7 and Windows Server 2008.

Measurement interval; WUP and Perfmon perform measurements with a one second interval, while computers process millions of instructions per second. Although we argue our measurements are sufficient for our purposes, we acknowledge the fact that data is lost with the instruments at hand.

Table IV
THE EC (IN JOULE) AND RUS FOR THE DOCUMENT GENERATOR SOFTWARE PRODUCT BEFORE AND AFTER CHANGING THE SOFTWARE.

	EC	RUS
Single-threaded	17,560	2,313.09
Multi-threaded	5,782	1,644.49

Room temperature; Of the three systems the server was the only one situated in a climate-controlled data center and as a consequence we can only guarantee identical conditions for this system. Although we tried to maintain consistency, we acknowledge the fact that, among others, room temperature could have influenced our measurements. We consider the insignificant differences found between measurements as a confirmation that the influence in our experiment was limited.

VII. CONCLUSION

In this paper we propose a metric to effectively communicate resource utilization measurements for a software product in relation to EC. The metric should be easy to understand, reported uniformly and clearly communicate key findings. We consider the viewpoints of multiple stakeholders wanting to address the sustainability of their software product through green software practices, and posed the following **research question**: ‘How can we effectively express the resource utilization for executing a software product in relation to the SEC?’. We provide an answer by constructing the RUS.

Following the goals of the of radar chart, the RUS delivers a single score based on selected dimensions and performance metrics. To calculate the RUS, the equation to calculate the surface of a radar chart was transformed into one that considers the relation between dimensions. A weight factor is added that enables stakeholders to determine the importance of each pair of dimensions. As the measurements can be expressed on a standardized scale they can be interpreted more easily, do not require knowledge on the individual metrics and can be compared between software applications. Additionally, the radar chart provides a means to visualize the measurements which helps to identify key findings.

Evaluating the RUS with two different datasets, showed that the RUS should be considered complementary to the EC and the execution time related to a software product. In general a lower RUS corresponds to a lower EC consumption figure, but with the server a case was also found where a lower RUS was accompanied by a higher EC. In these situations a trade-off should be made, like with quality attributes, favoring the aspect that is considered more important in a specific context. A limitation of the RUS found in its inability to be compared across systems of different classes.

Based on the work presented in this paper, we identify several direction for future research. First is to investigate the RUS more thoroughly. For example, the RUS could be used to compare between systems within the same class. Also a (standardized) means to determine the weight factor could aid in the RUS’ acceptance. A second direction is to investigate the positioning of the RUS in relation to more the generic eco-labels for the ICT domain. A final direction is to use RUS to create awareness on green software during the development process. By showing the impact of software development activities, software developers are enabled to address sustainability issues that might arise.

Acknowledgment

We would like to thank Fabiano Dalpiaz and Garm Lucassen and the reviewers of ICT4S for their valuable feedback to improve the paper.

References

[1] S. Murugesan, “Harnessing green it: Principles and practices,” *IT Professional*, vol. 10, no. 1, pp. 24–33, Jan 2008.

[2] Y. Sun, Y. Zhao, Y. Song, Y. Yang, H. Fang, H. Zang, Y. Li, and Y. Gao, “Green challenges to system software in data centers,” *Frontiers of Computer Science in China*, vol. 5, no. 3, pp. 353–368, 2011.

[3] K. Grosskop and J. Visser, “Identification of application-level energy optimizations,” in *Proceedings of ICT for Sustainability (ICT4S)*. Atlantis Press, 2013, pp. 101–107.

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

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

[6] S. Jansen, S. Brinkkemper, J. Souer, and L. Luinenburg, “Shades of gray: Opening up a software producing organization with the open software enterprise model,” *Journal of Systems and Software*, vol. 85, no. 7, pp. 1495–1510, 2012.

[7] C. Pang, A. Hindle, B. Adams, and A. E. Hassan, “What do programmers know about the energy consumption of software?” *PeerJ PrePrints*, vol. 3, p. e1094, 2015.

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

[9] E. A. Jagroep, J. M. E. M. van der Werf, R. Spauwen, L. Blom, R. van Vliet, and S. Brinkkemper, “An energy consumption perspective on software architecture,” in *Software Architecture*, ser. LNCS, no. 9278. Springer, 2015, pp. 239–247.

[10] E. A. Jagroep, J. 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.

[11] F. Chasin, “Sustainability: Are we all talking about the same thing state-of-the-art and proposals for an integrative definition of sustainability in information systems,” in *Proceeding of ICT for Sustainability (ICT4S)*. Atlantis Press, 2014, pp. 342–351.

[12] 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.

[13] 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.

[14] G. Procaccianti, P. Lago, and G. A. Lewis, “Green architectural tactics for the cloud,” in *Software Architecture (WICSA), 2014 IEEE/IFIP Conference on*, April 2014, pp. 41–44.

[15] G. Kalaitzoglou, M. Bruntink, and J. Visser, “A practical model for evaluating the energy efficiency of software applications,” in *Proceedings of ICT for Sustainability (ICT4S-14)*. Atlantis Press, 2014.

[16] C. Sahin, M. Wan, P. Tornquist, R. McKenna, Z. Pearson, W. G. J. Halfond, and J. Clause, “How does code obfuscation impact energy usage?” *Journal of Software: Evolution and Process*, 2016.

[17] 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.

[18] N. Rozanski and E. Woods, *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley, 2011.

[19] S. Barlowe and A. Scott, “O-charts: Towards an effective toolkit for teaching time complexity,” in *Frontiers in Education Conference (FIE), 2015. 32614 2015. IEEE*, Oct 2015, pp. 1–4.

[20] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, “Virtual machine power metering and provisioning,” in *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC ’10. New York, NY, USA: ACM, 2010, pp. 39–50.

[21] C. Ebert and S. Brinkkemper, “Software product management - an industry evaluation,” *Journal of Systems and Software*, vol. 95, no. 0, pp. 10 – 18, 2014.

[22] A. Kipp, T. Jiang, M. Fugini, and I. Salomie, “Layered green performance indicators,” *Future Generation Computer Systems*, vol. 28, no. 2, pp. 478 – 489, 2012.

[23] K. Lundfall, P. Grosso, P. Lago, and G. Procaccianti, “The green practitioner: A decision-making tool for green ict,” in *Proceedings of ICT for Sustainability (ICT4S)*. Atlantis Press, 2015, pp. 74–81.

[24] D. Magalhães, R. N. Calheiros, R. Buyya, and D. G. Gomes, “Workload modeling for resource usage analysis and simulation in cloud computing,” *Computers & Electrical Engineering*, vol. 47, pp. 69–81, 2015.

[25] M. Poess and R. O. Nambiar, “Energy cost, the key challenge of today’s data centers: a power consumption analysis of tpc-c results,” *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1229–1240, 2008.

[26] K. Singh, M. Bhadauria, and S. A. McKee, “Real time power estimation and thread scheduling via performance counters,” *ACM SIGARCH Computer Architecture News*, vol. 37, no. 2, pp. 46–55, 2009.

[27] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing,” *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755 – 768, 2012, special Section: Energy efficiency in large-scale distributed systems.

[28] R. Koller, A. Verma, and A. Neogi, “WattApp: an application aware power meter for shared data centers,” in *Proceedings of the 7th international conference on Autonomic computing*. ACM, 2010, pp. 31–40.

[29] P. Somavat, V. Namboodiri *et al.*, “Energy consumption of personal computing including portable communication devices,” *Journal of Green Engineering*, vol. 1, no. 4, pp. 447–475, 2011.

[30] R. Carpa, O. Gluck, L. Lefevre, and J.-C. Mignot, “Improving the energy efficiency of software-defined backbone networks,” *Photonic Network Communications*, vol. 30, no. 3, pp. 337–347, 2015.

[31] J. Espadas, A. Molina, G. Jimnez, M. Molina, R. Ramrez, and D. Concha, “A tenant-based resource allocation model for scaling software-as-a-service applications over cloud computing infrastructures,” *Future Generation Computer Systems*, vol. 29, no. 1, pp. 273 – 286, 2013, including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.

[32] E. Kern, M. Dick, S. Naumann, and A. Filler, “Labelling sustainable software products and websites: Ideas, approaches, and challenges,” in *Proceedings of ICT for Sustainability (ICT4S)*. Atlantis Press, 2015, pp. 82–91.

[33] G. Procaccianti, P. Lago, and G. A. Lewis, “A catalogue of green architectural tactics for the cloud,” in *Maint. and Evol. of Service-Oriented and Cloud-Based Systems (MESOCA), 2014 IEEE 8th Int’l Symp. on the*, Sept 2014, pp. 29–36.

[34] H. Schütz, S. Speckesser, and G. Schmid, “Benchmarking labour market performance and labour market policies: Theoretical foundations and applications,” WZB Discussion Paper FS I 98-205, 1998. [Online]. Available: http://hdl.handle.net/10419/43918

[35] H. Mosley and A. Mayer, “Benchmarking national labour market performance: A radar chart approach,” WZB Discussion Paper FS I 99-202, 1999. [Online]. Available: http://hdl.handle.net/10419/43952

[36] G. Bekaroo, C. Bokhoree, and C. Pattinson, “Power measurement of computers: analysis of the effectiveness of the software based approach,” *Int. J. Emerg. Technol. Adv. Eng*, vol. 4, no. 5, pp. 755–762, 2014.

[37] T. Vogelsang, “Understanding the energy consumption of dynamic random access memories,” in *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, Dec 2010, pp. 363–374.

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

[39] 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.

[40] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln, *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

A.2.2 Awakening Awareness on Energy Consumption in SE

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 processes 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 to be a key driver for the energy consumption of Information and Communication Technology (ICT) solutions by academia [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, sustainabilty becomes a true 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 to 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 able to adopt solutions provided by research.

With the growing attention for corporate social responsibility, Software Product 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 introduced 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 products, we then measured how the awareness developed over several sprints. To measure the development of the awareness we created a specialized awareness model for SEC, inspired on the work of [19], that served as a basis for the surveys held with the stakeholders after each sprint.

The main contributions of this paper are twofold:

- **Awareness model for SEC:** The model 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 into the areas that require extra attention in relation to the SEC and ECP.

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

The paper is structured as follows: The next section presents our research questions (Section II) 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 in the introduction, our study was structured around the following **main research question**:

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' which 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 stakeholder 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: 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, development efforts applying green practices [15] and selecting the right 'Collection types' [20] show reduced energy consumption up to respectively 25% and 300%. If we position sustainability as a quality property for the software [3], [5] we can go back even 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 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 to 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, which is bound to affect their practice [30]. Following the economic dimension of sustainability [3], software-oriented data analytics [25] provide 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 prefer 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].

IV. Research Design

To answer our main research question we have conducted an embedded multiple-case study [32], where we measure

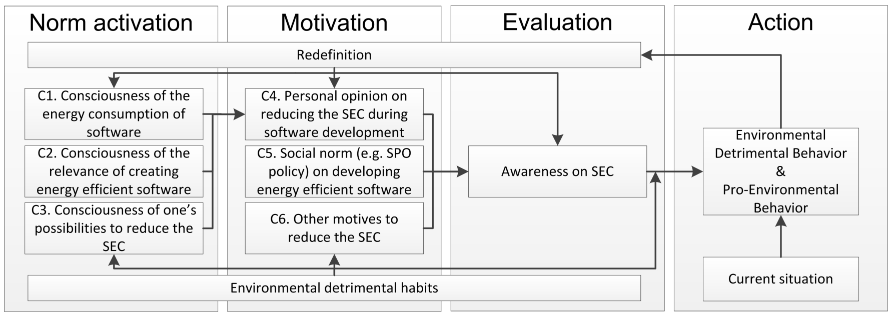

Norm activation
Motivation
Evaluation
Action
Redefinition
C1. Consciousness of the energy consumption of software
C2. Consciousness of the relevance of creating energy efficient software
C3. Consciousness of one's possibilities to reduce the SEC
C4. Personal opinion on reducing the SEC during software development
C5. Social norm (e.g. SPO policy) on developing energy efficient software
C6. Other motives to reduce the SEC
Awareness on SEC
Environmental Detrimental Behavior & Pro-Environmental Behavior
Current situation
Environmental detrimental habits

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

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 the energy consumption on the account of the software product, i.e. its SEC.

A prerequisite for valid measurements 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 processing.

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, 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.

[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.

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).

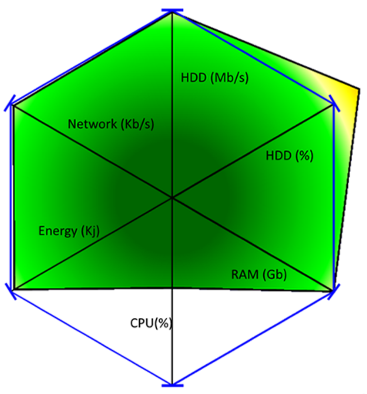
In particular, the *radar chart* graphically shows the RUS [18] and is meant to enhance the communication of the measurements to stakeholders and highlight key findings. As adopted in [12], visual information coveys familiar indicators and gauges, e.g. percentages and bar charts, that 'non-energy experts' 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 organisation 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 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**



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.

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 limited population size of our dataset (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.

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].

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

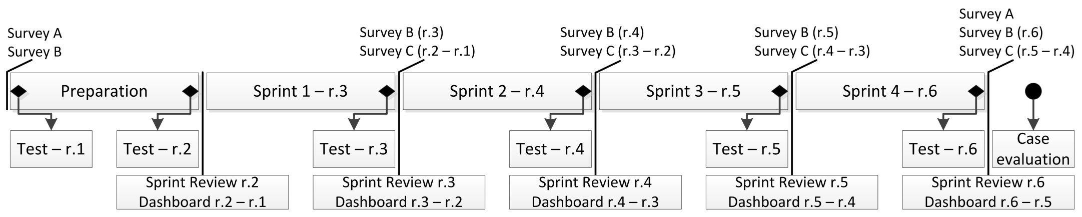


Figure 3. The case study organization including sprints, test periods, sprint reviews, energy dashboard presentations and surveys.

A. *Study Execution*

With DG we deviated from the case description by excluding release 9.0.0 from the study. Even with help from 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 were not able compensate for this exclusion, resulting in a gap with the data 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. Compared to DG the inability to physically attend sprint review meetings resulted in 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). Consequently, the RS scores were calculated based on the survey results of twelve team members. For the energy dashboard we managed to perform nine runs per release, instead of the required ten, due to sharing of the test resources and issues with Windows Performance Monitor. The available resources for load testing were also used for production testing which simply had a higher priority. Investigation with Performance Monitor pointed out the tool automatically deletes data when the available hard disk space is limited, an issue we solved with a simple reconfiguration. However, despite missing one run we still obtained sufficient data per release to produce valid energy dashboards.

The case 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. The results of DG (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, the scores of the DG stakeholders (Fig. 6) are increasingly negative over sprint reviews – the total score changes from 14 to -24. The only exception to the negative acceptance is with the effort expectancy (EE) construct where score remains +5, implying that the dashboard is user friendly. The attitude towards technology (AT) follows in a second place with a score moving from +6 to 0. Scores concerning the behavioral intention (BI), performance expectancy (PE) and social influence (SI) all become increasingly negative, +8 to -11, -3 to -5 and -2 to -13 respectively, with the only difference being that the BI scores start out positive.

For RS (Fig. 7), the total score shifts from -79 to -118 and we again only find positive scores with the EE construct (+6). 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 just presented, 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 with DG (Fig. 4) switched from negative (-2) to positive (+5), indicating an increased willingness to determine energy consumption aspects in relation to software in general, and became more negative (-13 to -18) with RS (Fig. 5). A change on this construct was expected given the fact that the stakeholders participated in a study on the topic. The case evaluations learned that both teams indeed were more aware of the SEC and would keep the topic in mind during development, hence implying a change in behavior. However, the main focus still remains on product functionality in line with strategy and customer demand.

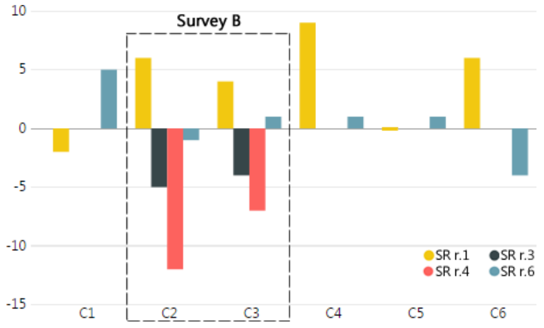


Figure 4. DG case: The awareness scores per construct over sprint reviews.

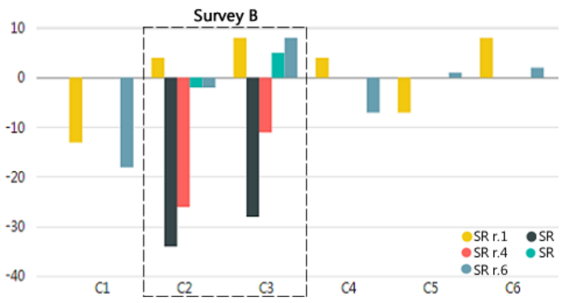


Figure 5. RS case: The awareness scores per construct over sprint reviews.

Software architecture: For C2 and C3, while stakeholders from both cases increasingly want to know the SEC of their software product, the RS results show disagreement with the influence of software on energy consumption being large and potentially explain the increased negativity with C1. This is 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 product. One RS developer explained previous experiences with mobile software and stated that the effectiveness of the efforts is clouded by the different layers of the software stack (e.g. operating system and middleware). The move towards a more neutral sentiment on this statement (-15 to -6) does indicate a greater acknowledgement of the role of software. However, hardware or other aspects, are still believed to have a greater energy consumption impact than the software.

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

Similarly, the results for C6 confirm the focus on software architecture. The stakeholders reconsidered the statement to include code optimizations to reduce SEC in the backlog (+6 to -3 with DG, and +11 to +6 for RS). A possible cause is that the stakeholders increasingly realized SEC is related to non-functional requirements of the software [4] which are typically not included in the backlog. Also the stakeholders disagreed with 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 for both cases, there seems to be an underestimation of the complexity involved with reducing the SEC. After the first dashboard presentation, the scores become negative with stakeholders realizing the complexity involved and move towards neutral or even positive after they have had more time to think of 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 this statement, possibly indicating that SEC is discussed by specific groups within the team instead of the entire team.

SEC on the team agenda: Given the developments with C1, C2 and C3, a decline with C4 was expected. The individual statements of DG show that a slight positive sentiment remains as the stakeholders are willing to address the SEC when allowed to spend time on this. However, again in line the discussion so far, RS becomes negative (+4 to -7). Nevertheless, the SEC has been put on the agenda and is kept in mind with other quality aspects of the software.

When asked whether extra attention towards this topic was required, stakeholders from both cases indicated this was not the case unless 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, we found that SEC is discussed by (groups of) stakeholders. Also both acknowledge that reducing the SEC benefits the organization and the customer, however customer demand and market trends are leading in determining the future of the product. These results indicate willingness to address the SEC when sustainability goals are in place, be it by the organization or by customers. However, even without a clear relation to strategic goals, an SPO could benefit from these results by promoting energy efficient software development. An overall reduction of the total cost of ownership for a product allows an SPO to, e.g., revise the pricing model to be more competitive.

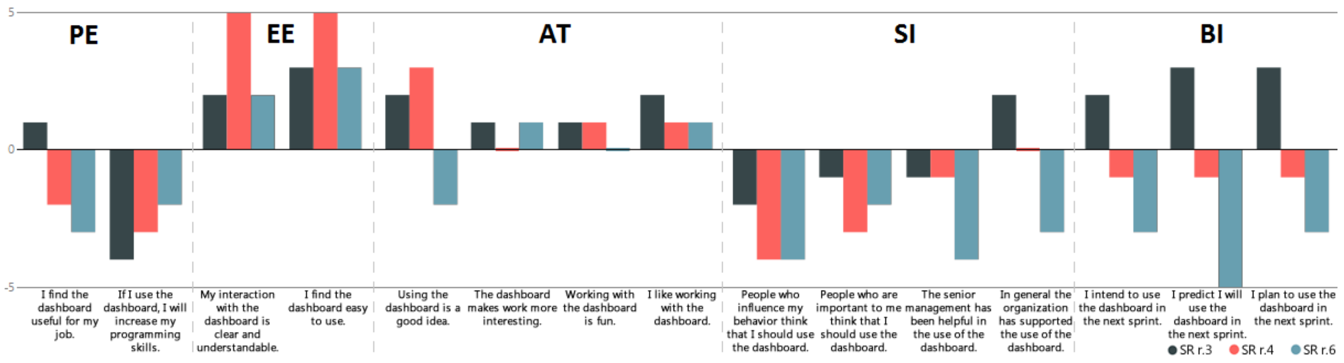


Figure 6. DG case: The scores on the individual acceptance statements (survey C), grouped per construct.

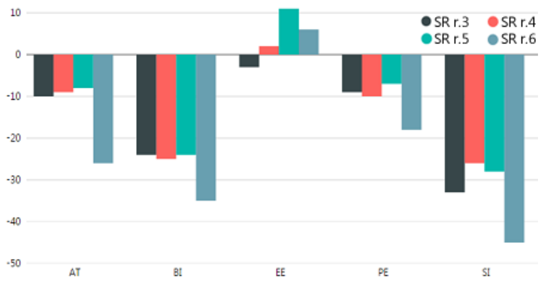


Figure 7. RS case: The acceptance scores per construct over sprint reviews.

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: By interpreting the scores in light of the awareness scores, we can only partially explain the negative sentiment by stakeholders not knowing how to reduce the SEC or by becoming more aware of the limitations imposed by, e.g., the technology used. In this light, showing 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 awakened awareness combined with software engineering knowledge on how to decrease SEC would be most effective.

Despite potentially being confronting, 'Using the dashboard is a good idea' (AT) is one of few statements that was positive for RS. Also DG stakeholders indicate they like working with the dashboard and the dashboard makes work more interesting, but they find using the dashboard not a good idea. While we cannot explain the unexpected decline with the latter, the former two potentially indicate that the dashboard is considered a good stimulus in this context.

Presentation frequency: Apart from being confronting, the frequency of presenting the dashboard, i.e. after every sprint, could also contribute to the negative sentiment. For example, while the statement concerning the usefulness for the job starts positive in both cases, consecutive measurements indicate disagreement. During the evaluation, stakeholders indicate that SEC (similar to other quality attributes) is to be considered periodically but not after every sprint. Unless there is a concrete motive to do so.

Target audience: with DG a contrasting finding is found with the PE statements: the dashboard is increasingly less useful for the job, but stakeholders disagree less with using the dashboard as a means to increase programming skills. With RS both statements are negative, although the first starts out positive. 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, showing that stakeholders find little support with others, including management, in using the energy dashboard, this could be a signal for an SPO that energy aware software development does have a place within the organization. However extra attention should be paid the information that is presented and to whom.

C. SQ3: incorporating SEC in the development process

The results show that the stakeholders are able to make weighted decisions with respect to SEC, or put otherwise, 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:

Sustainability goals: Stakeholders indicate a reduced energy consumption to be beneficial for both the organization and the customer. Positioning reducing SEC as a strategic goal for the organization helps stakeholders to justify efforts addressing the SEC of their products. By embedding SEC in the organization, an SPO can also experience benefits from the social factors that motivate stakeholders. An effect that can be potentially amplified by means of, e.g., gamification.

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 a 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, strengthening the relation with sustainability goals.

Stimulus availability: Despite the limited acceptance, the energy dashboard proved useful to show the effects of development efforts. Quantifying helps making SEC more concrete, hence enabling informed decision making. Additionally, the energy dashboard can be used to put specific achievements in the spotlight to encourage such efforts. With RS, for example, the reduced CPU utilization by solving a long term bug (see Fig. 2) was put in the spotlight using the energy dashboard. When an energy dashboard, or a different 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, as agile has become an industry standard, that RS and DG are representative for the software industry in general. Further investigation is still 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 organisation 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 multiple-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 *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. 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 *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ser. ICSE '15. Piscataway, NJ, USA: 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 Publishing Company, Incorporated, 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.