Energy-Efficient Software

Giuseppe Procaccianti

May 11th, 2015

SIKS Dissertation Series No. 2015-13
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

This work has been partially sponsored by the European Fund for Regional Development under project MRA Cluster Green Software.

ISBN 978-94-6295-150-1

Cover design by Giacomo Marseglia and Uitgeverij BOXPress
Cover image credits to Le Moal Olivier / 123RF Stock Photo
Published by Uitgeverij BOXPress, ‘s-Hertogenbosch
Typeset in LaTeX by the author

VRIJE UNIVERSITEIT

Energy-Efficient Software

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. F.A. van der Duyn Schouten,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de Faculteit der Exacte Wetenschappen
op maandag 11 mei 2015 om 11.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Giuseppe Procaccianti

geboren te Palermo, Italië

promotoren:	prof.dr.	P. Lago
	prof.dr.	M. Morisio

Contents

1 Introduction

1.1 The Unsustainable ICT

The environmental impact of Information and Communication Technologies (ICT) is constantly growing at a brisk pace. A report issued by the Global e-Sustainability Initiative [75] shows that the greenhouse gas (GHG) emissions of the ICT sector are projected to rise to 1.3 GtCO2e (2.3% of global emissions) by 2020, with a growth of 3.8% from 2011. In particular, datacenters are the fastest growing category in ICT emissions, with a 7.1% annual rate.

According to Koomey [66], electricity used in global data centers in 2010 was estimated between 203.4 and 271.8 TWh, that is between 1.1% and 1.5% of total worldwide electricity use, respectively. For the US that number was between 1.7 and 2.2%. In addition to those figures, the rapid adoption of Cloud Computing technologies will increase the demand for data centers in the next future, as reported by Greenpeace [44]. In the only 6 years between 2007 and 2012, the global ICT electricity consumption increased by more than 200 TWh, going from 4 to 4.7% of the world total (see Figure 1.1).

Putting aside the environmental sustainability aspects, economic sustainability concerns are also rising. As energy costs increase due to depletion of traditional energy sources, the Total Cost of Ownership (TCO) of ICT infrastructures will become prohibitive for many companies.

This tells us that the energy efficiency of ICT has to improve. The responsibility of dealing with this issue undoubtedly falls upon ICT professionals and researchers.

To date, most of the achieved improvements in ICT energy efficiency are hardware-related. Only in the very last years, software technologies are being considered for energy optimization. To some extent, this has an analogy with the history of Software Engineering: while at the beginning of the Information Era software was undistinguished from hardware in the eyes of IT professionals,

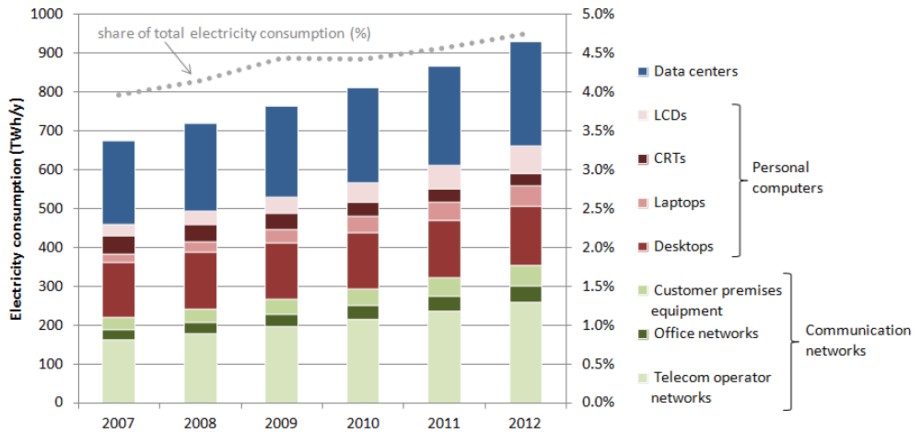


Figure 1.1: Worldwide electricity consumption of communication networks, personal computers and data centers. [76].

nowadays its complexity and abstraction require a dedicated discipline and an engineering approach. This is precisely the reason why we consider software energy efficiency as a Software Engineering problem. Recently, the research community [74] has recognized the potential of energy efficiency in Software Engineering and defined its grand challenges. Some of those challenges are the object of study of this thesis.

1.2 The Quest for Energy-Efficient Software

In 1995, Niklaus Wirth stated what became famous as the "Wirth's law" [136]:

"Software is getting slower more rapidly than hardware becomes faster."

This observation is sometimes seen as the counterpart of Moore's law [87] in the sense that hardware improvements in terms of performance are negated by software inefficiencies. The reason lies in the fact that as hardware resources (e.g. CPU, memory, storage) become cheaper, software designers and developers are not concerned anymore with writing software that makes an efficient use of those resources. On the contrary, for market reasons, it is more rewarding for them to provide more features in their products (thus increasing complexity) [136].

This phenomenon has inevitable repercussions on energy consumption. In Figure 1.2 you see how the energy efficiency of microprocessor-based computer devices has increased during the years, when compared to computational power[1]. However, consider these figures with the trend of the total energy consumption of ICT, as shown in Figure 1.1. Although hardware devices consume less energy per computation, the overall ICT electricity consumption still increases.

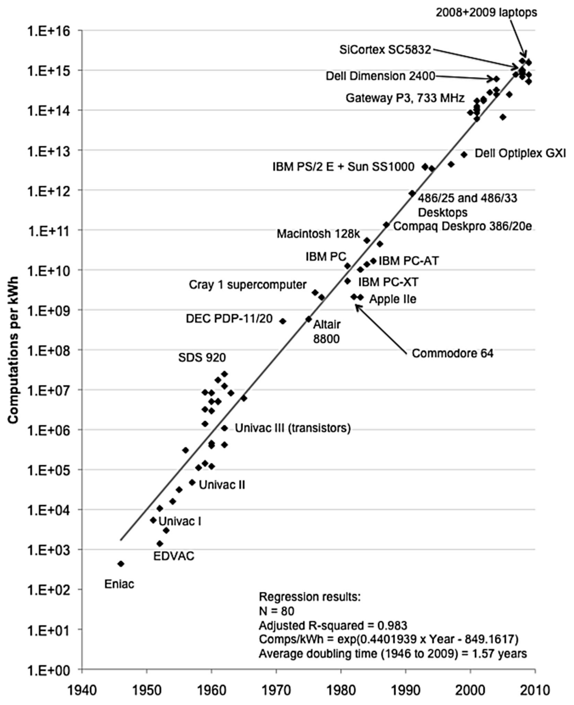


Figure 1.2: Computations per kilowatt-hour over time [65].

Clearly, this is also due to the fact that the total number of devices has increased substantially: creating more efficient equipment leads to a decrease in the price of the provided service or product, which in turn increases the demand – a phenomenon known as *rebound effect* [14].

But this is just part of the reason. Modern hardware devices also have more fine-grained energy management capabilities: multiple power states, sleep modes and hibernation are common features of end-user equipment. These capabilities, however, impair *energy proportionality*, i.e. the ability of the system to consume energy proportionally with its load [127]. For example, a perfectly

[1]In the figure, “computation” is expressed as a normalized index of addition time, where human performance equals to 1.

energy-proportional system at 100% load would consume five times as much as at 20% load. Currently, this ratio is much higher. Moreover, as confirmed by the empirical evidence presented later in this thesis, the percentage increase of power consumption with respect to idle is much higher on modern hardware devices than on their technological predecessors.

Inefficient software technologies, i.e. that waste hardware resources, impact energy consumption even more than system load and performance. Regardless of how much we improve the hardware, without energy-efficient software the ICT energy consumption will continue to rise.

1.3 Research Questions

As just discussed, software plays an important role in the energy consumption of ICT. However, the current State-of-the-Art of Software Engineering does not provide consolidated knowledge on the deep and complex relationship between software and energy consumption. This thesis aims at exploring this relationship from a *software-centric* perspective.

To do so, the abstraction layers of computing devices are traversed in a bottom-up fashion: first, we analyze the correlation between energy and software applications, looking for patterns and mechanisms that affect energy consumption. Then, we look at how software development can influence these mechanisms, assessing the impact of coding practices. Subsequently, we scale up to the architectural level, to discover whether energy efficiency can be addressed at the early stages of software design.

Ultimately, our goal is to provide an approach to engineer energy-efficient software, both at architecture, design and code level.

The related main Research Question for this thesis can be stated as:

RQ: How can we engineer energy-efficient software?

This RQ can be decomposed into more specific questions, to be addressed with the methods exposed in the next paragraph.

- Energy is ultimately consumed by hardware resources, used by software. It is then needed to understand which resources are responsible of consuming more energy during software execution. This allows to identify the most promising strategies for optimization. More importantly, if a significant correlation is found between consumed energy and software behavior in terms of usage of resources, it is possible to estimate energy consumption only from run-time software execution metrics.

RQ 1. What is the correlation between software properties and hardware energy consumption?

  - *RQ 1a. Is hardware resource usage correlated with energy consumption during software execution?*
  - *RQ 1b. How can software properties be used as a predictor for hardware energy consumption?*

- Once we discover the mechanisms that characterize the relationship between software behavior and energy consumption, the next step is guiding developers into creating energy-efficient software or improving the energy efficiency of existing software applications. For this purpose, a number of best practices and guidelines have been suggested, mostly in industrial literature. We have to assess whether the available best practices and guidelines for energy efficient software have a quantifiable impact, by empirically validating them in a controlled environment.

RQ 2. What is the impact of using best practices for software energy efficiency?

- Traditional software quality aspects (QAs, e.g. security, reliability, maintainability) are commonly addressed at a software architectural level. This is because qualities result from seemingly independent software properties, determined by architectural design decisions taken at early stages of software development [12]. Software architecture captures these decisions and contextualizes them, facilitating the analysis of quality aspects and providing reusable solutions to address them. If energy efficiency is a proper QA, we must be able to analyze it at the architectural level. For this reason, we have investigated existing large-scale software architectures that consider energy efficiency as a main concern. By doing so, we aim at discovering reusable solutions to address energy efficiency at architectural level.

RQ 3. How can software architectural solutions realize energy efficiency?

  - *RQ 3a. Are there software architectural solutions that address energy efficiency aspects?*
  - *RQ 3b. How can architectural solutions for energy efficiency be made reusable?*

- From our bottom-up approach, we have been able to identify software-hardware mechanisms, coding guidelines and reusable architectural solutions for software energy efficiency. However, these results need to be generalized and put into a bigger picture, in order to deliver reusable knowledge for practitioners and other researchers. For this purpose, a conceptual framework is needed, that encapsulates all these elements into high-level *strategies* for software energy efficiency i.e. broad, long-term design approaches, composed of different tactics addressing multiple levels of abstraction.

RQ 4. Can we provide strategies to improve software energy efficiency?

1.4 Research Methods

The relationship between hardware and software is complex, with technologies acting as confounding factors, such as distributed systems, virtualization, mobile computation, cloud computing, etc. Our assumption, when addressing the problem of software energy efficiency, is that it is an *emergent* property of software systems in use: the complexity of the hardware-software interactions and the multiple software layers create an environment that we are currently unable to deterministically describe.

Consequently, in this thesis we adopt an *inductive* approach, i.e. we build knowledge on software energy efficiency by gathering and analyzing empirical evidence [9]. For this purpose, both quantitative and qualitative analysis techniques were used. In particular:

- *Systematic Literature Review (SLR).* This qualitative research method is defined by Kitchenham et al. [64] as "a form of secondary study that uses a well-defined methodology to identify, analyse and interpret all available evidence related to a specific research question in a way that is unbiased and (to a degree) repeatable". This method is extremely useful in establishing background knowledge, but also to evaluate the degree of maturity of a certain field. We adopted this method to identify the state-of-the-art of energy efficiency in Cloud-based software architectures.
- *Literature Review.* This qualitative research method differs from the SLR in that it is less formal and structured, but gives also more freedom in selecting relevant sources and collecting evidence. When there is no need of representing the state-of-the-art extensively and systematically, but we want to provide background information, a literature review is a more efficient choice. We used this method to provide an overview of software energy measurement and modeling methods and tools.
- *Quasi-experiment.* This quantitative empirical enquiry is based upon the manipulation of one factor (or variable) in a controlled setting. Unlike randomized controlled trials, where treatments are assigned to different sub-

jects through randomization, in quasi-experiments the assignment is done using a specific criterion. In software engineering, quasi-experiments are more common than randomized trials, mostly for feasibility reasons [57]. In our case, this choice is motivated by the nature of our subjects (software applications) and treatments (usage scenarios, development practices) which makes randomization unpractical and sometimes not meaningful. Our quasi-experiment design follows the process and guidelines provided by Wohlin et al. [138].

1.5 Thesis at-a-Glance

Figure 1.3 gives an overview of this thesis, showing how the RQs relate to the ultimate goal of this work.

Quasi-empirical experiments were designed and performed to answer RQ1a and RQ2, as their lower level of abstraction makes experimentation feasible. RQ1b was investigated via reviewing the literature on software power measurement and modeling. RQ3 is focused on the architectural level, in particular on large-scale, Cloud-based software applications. Performing an empirical experimentation on such a scale proved to be unfeasible, thus we adopted a different research method: a secondary study on existing architectural solutions that address energy efficiency. This analysis was conducted by means of a systematic literature review (SLR). From the results of the SLR, we were able to identify the stakeholders involved in software energy efficiency, as well as architectural strategies that can be used to address energy efficiency aspects. From the gathered empirical evidence, we answer RQ4 by analyzing our results and synthesizing them into holistic strategies for software energy efficiency. The outcome of this synthesis activity is a conceptual framework to engineer energy-efficient software.

1.6 Outline of Thesis and Publications

This dissertation is composed of the following chapters:

- *Chapter 2.* This chapter answers RQ 1 by means of a twofold contribution: first, it presents the design and results of an experiment on the impact of software on energy consumption. Secondly, it provides a literature survey on software power measurement and modeling. The aim is to give a background on the relationship between software and hardware.

  Parts of this chapter have been previously published as:

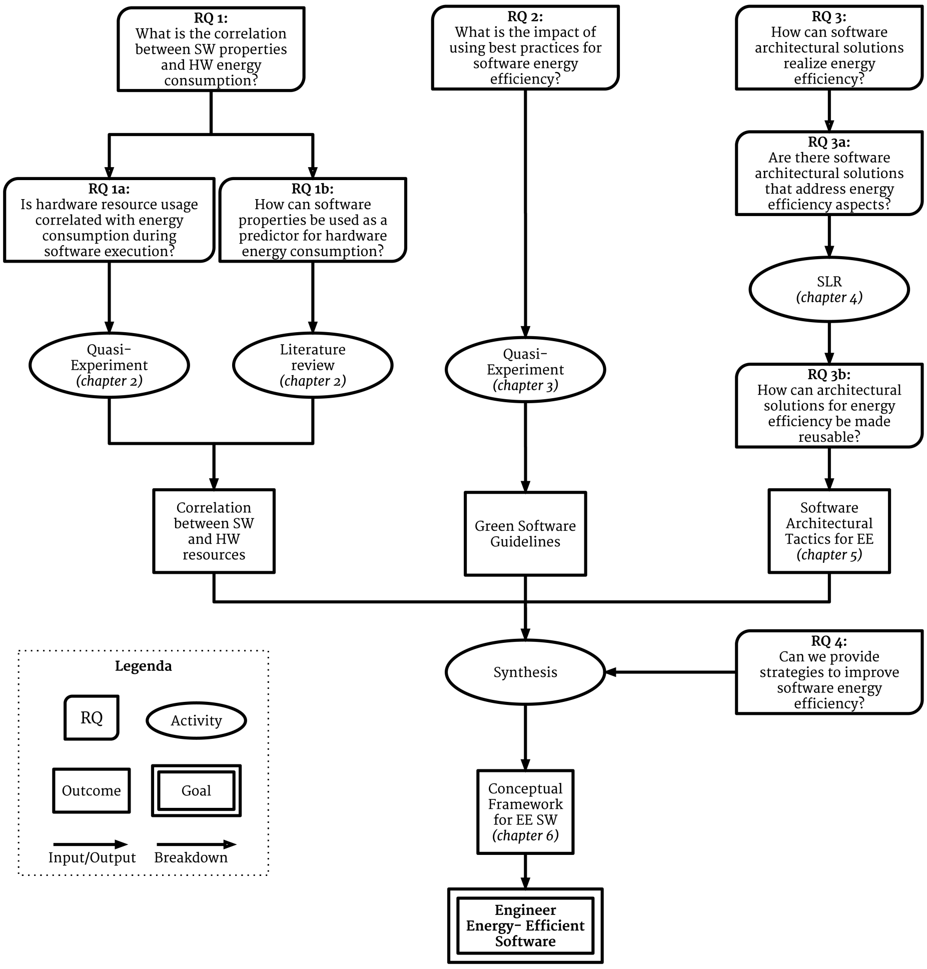

RQ 1:
What is the correlation between SW properties and HW energy consumption?
RQ 2:
What is the impact of using best practices for software energy efficiency?
RQ 3:
How can software architectural solutions realize energy efficiency?
RQ 1a:
Is hardware resource usage correlated with energy consumption during software execution?
RQ 1b:
How can software properties be used as a predictor for hardware energy consumption?
RQ 3a:
Are there software architectural solutions that address energy efficiency aspects?
SLR (chapter 4)
Quasi-Experiment (chapter 2)
Literature review (chapter 2)
Quasi-Experiment (chapter 3)
RQ 3b:
How can architectural solutions for energy efficiency be made reusable?
Correlation between SW and HW resources
Green Software Guidelines
Software Architectural Tactics for EE (chapter 5)
Legenda
RQ
Activity
Outcome
Goal
Input/Output
Breakdown
Synthesis
RQ 4:
Can we provide strategies to improve software energy efficiency?
Conceptual Framework for EE SW (chapter 6)
Engineer Energy- Efficient Software

Figure 1.3: Overview of the RQs of this thesis work, along with performed activities and outcomes.

– Procaccianti G., Vetrò A., Ardito L., Morisio M. (2011)
Profiling Power Consumption in Desktop Computer Systems. In proceedings of: *Information and Communication on Technology for the Fight against Global Warming (ICT-GLOW) 2011.* Toulouse, France. Ed. Springer.

Personal contribution: as main author, I conducted the empirical experimentation and performed the data analysis. The paper was mainly written by the first, second and third author. The fourth author provided detailed reviews.

  - Procaccianti G., Ardito L., Vetrò A., Morisio M. (2012)
    Energy Efficiency in the ICT - Profiling Power Consumption in Desktop Computer Systems. In: *Energy Efficiency - the Innovative Ways for Smart Energy, the Future Towards Modern Utilities.* Ed. Prof. Moustafa Eissa, Helwan. Intech
    Personal contribution: as main author, I conducted the empirical experimentation and performed the data analysis. The paper was mainly written by the first and second author. The third and fourth author provided detailed reviews.

- *Chapter 3.* In this chapter, we present the design and results of an empirical experiment to assess the impact of industrial practices for energy-efficient software development. This allows us to answer RQ 2.

  This chapter has been submitted as:

  - Procaccianti G., Fernandez, H., Lago, P. (2014)
    Empirical Evaluation of Best Practices for Energy-Efficient Software Development. Submitted to: *IEEE Transactions on Software Engineering*
    Personal contribution: Together with the second author, I conducted the empirical experimentation and performed the data analysis. The paper was written and reviewed by all of the authors.

- *Chapter 4.* In this chapter, we present the design and results of a Systematic Literature Review on energy efficiency in Cloud Software Architectures. This study answers RQ 3a.

  This chapter has been published as:

  - Procaccianti G., Bevini, S., Lago, P. (2013)
    Energy Efficiency in Cloud Software Architectures. In proceedings of: *27th Conference on Environmental Informatics (ENVIROINFO 2013).* Hamburg, Germany. Ed. Shaker-Verlag.
  - Procaccianti G., Lago, P., Bevini, S. (2014)
    A systematic literature review on Energy Efficiency in Cloud Software Architectures. *Sustainable Computing (SUSCOM)* - Special Issue on Software Engineering Aspects of Green Computing (SEAGC)
    Personal contribution: Together with the third author, I conducted the systematic review and performed the data analysis. The papers

were mainly written by the first author and reviewed by all of the authors.

- *Chapter 5.* In this chapter, we reflect on the results of the SLR and package them into a set of architectural tactics for energy efficiency. This contribution answers RQ 3b.

  This chapter has been previously published as:

  - Procaccianti G., Lago, P., Lewis, G.A. (2014)
    Green Architectural Tactics for the Cloud. In proceedings of the *11th Working IEEE/IFIP Conference on Software Architecture (WICSA 2014).* Sydney, Australia. Ed. IEEE.
  - Procaccianti G., Lago, P., Lewis, G.A. (2014)
    A Catalogue of Green Architectural Tactics for the Cloud. In proceedings of the *IEEE 8th Symposium on the Maintenance and Evolution of Service-Oriented Systems and Cloud-Based Environments (MESOCA 2014).* Victoria, Canada.
    Personal contribution: as main author, I developed the tactics and documented them. The papers were written and reviewed by all of the authors.

- *Chapter 6.* In this chapter, we build upon the results obtained so far and we present a conceptual framework to engineer energy-efficient software. This contribution answers RQ 4.

  This chapter has been previously published as:

  - Ardito L., Procaccianti G., Vetrò A., Torchiano M. (2014)
    Understanding Green Software Development: A Conceptual Framework. In *IT Professional*, pp.1-6, IEEE.
    Personal contribution: The framework was developed by all of the authors. I personally wrote Part 3 of the contribution. The paper was reviewed and revised by all of the authors.

- *Chapter 7.* In this chapter, we present an approach to systematically identify energy efficiency issues (*hotspots*) in software applications, starting from the knowledge base we built. The approach is an example of a strategy to improve software energy efficiency, hence related to RQ 4. However, the approach has not been properly validated yet, thus we cannot claim it answers the research question. We present it here as a preliminary step for our future works.

  This chapter has been published as:

– Procaccianti G., Lago P., Vetrò A., Mendez Fernandez, D., Wieringa, R. (2014)
The Green Lab: Experimentation in Software Energy Efficiency. In: Proceedings of the 37th International Conference on Software Engineering (ICSE 2015).
Personal contribution: The approach was developed by the first, second, third and fourth authors. The paper was reviewed and revised by all of the authors.

2 Background: Software and Energy

This chapter answers RQ 1 by means of a twofold contribution: first, it provides empirical evidence to provide a background on the relationship between software and hardware. For this purpose, an experiment has been designed, consisting in running benchmarks on two common desktop machines, simulating typical scenarios and then measuring the energy consumption and resource usage to extract indicative figures. Secondly, it provides a literature review on software power measurement and modeling. In spite of its relative immaturity, the state of the art in energy efficient software engineering already yields a rich set of reusable techniques to measure and model software energy consumption. As part of our research effort, we survey this body of knowledge, which as such can be used as a reference for selection.

2.1 Profiling Software Power Consumption

In this section we presents the design and results of an experiment, consisting in running benchmarks[1] on two common desktop machines.

This Section is organized as follows:

- Subsection 2.1.1 describes the experiment design process in all of its steps;
- in Subsection 2.1.2 we present the results of the experiment;
- in Subsection 2.1.3 we discuss the results in detail, providing additional insights.

[1] *A computer benchmark is typically a computer program that performs a strictly defined set of operations (a workload) and returns some form of result (a metric) describing how the tested computer performed.* [81] In our benchmark the *workload* is a set of usage scenarios and the *metric* is the power consumption.

2.1.1 Study Design

Goal Description

Our experiment design begins by describing our experimental goal. The goal is defined through the Goal-Question-Metric (GQM) approach [8]. In Table 2.1, we present the goal, research question and metrics we used in this study.

Goal	*Evaluate*	hardware resource usage
	for the purpose of	determining their influence
	with respect to	energy consumption
	from the viewpoint of	the user
	in the context of	software applications
Question	Is hardware resource usage correlated with energy consumption?	
Metric	CPU Usage (percentage)	
Metric	Memory Usage (reads/writes)	
Metric	Disk Usage (reads/writes)	
Metric	Network Usage (Packets/sec)	
Metric	Consumed Power (Watts)	

Table 2.1: The GQM Model

From the goal we derive RQ 1a, namely:

RQ1a: Is hardware resource usage correlated with energy consumption during software execution?

The research question asks for a quantifiable relationship between power consumption and actual usage of the hardware, by selecting four metrics relative to the main resources (CPU, Memory, Disk and Network) and one metric related to power consumption, namely the readings of the consumed power by the test machine obtained from an external power meter (see Instrumentation).

Variable Selection

In order to answer the Research Question, it is necessary to specify the independent variables that will characterize the experiment. In this study, the independent variables are represented by 11 usage scenarios. The scenarios represent a sample of the study population, i.e. common operations for a desktop user. They provide benchmarks (see Section 1) for the different resources of the computer system. Our usage scenarios are described in detail in the remainder of this section.

0 - Idle. This scenario evaluates power consumption during idle states. In order to reduce confounding factors, most of the automatic services of the OS were disabled (i.e. Automatic Updates, Screen Saver, Anti-virus and such).

1 - Web Navigation. This scenario represents one of the most common activities for a basic user - Web Navigation. During the simulation, the user starts a web browser, inputs the URL of a web page and follows a determined navigation path. Google Chrome has been chosen as the browser for this scenario because of its better performance on the test system, which allowed us to increase navigation time. The website selected for this scenario is the homepage of the SoftEng research group `http://softeng.polito.it`, which is managed by the authors of this study. This allows to keep constant the contents and navigation path during all the scenario runs.

2 - E-Mail. This scenario simulates sending and receiving E-Mails. For this scenario's purpose, a dedicated E-Mail account has been created in order to send and receive always the same message. In this scenario, the user opens an E-Mail Client, writes a short message, sends it to himself, then starts checking for new messages by pushing on the send/receive button. Once the message has been received, the user reads it (the reading activity has been simulated with an idle period), then deletes the messages and starts over.

3 - Productivity Suite. This scenario evaluates power consumption during the usage of interactive applications such as productivity suites. For this scenario, Microsoft Word 2007 has been selected, as it is one of the most used Word Processors. During the scenario execution, the user starts the application and creates a new document, filling it with content and applying several text editing/formatting functions, such as enlarge/shrink Font dimension, Bold, Italics, Underlined, Character and background colors, Text alignment and interline, lists. Then the document is saved on the machine's hard drive. For each execution a new file is created. The old file gets deleted after each scenario execution.

4 - Data Transfer (Disk). This scenario evaluates power consumption during File System operations, namely the displacement of a file over different positions of the hard drive, which is a very common operation. For this scenario's purpose, a data file of a relevant size (almost 2 GB) has been prepared in order to match the file transfer time with the prefixed scenario duration (5 minutes). The scenario structure is as follows: the system user opens an Explorer window, selects the file and moves it to another location. It waits for file transfer to end, then closes Explorer and exits.

5 - Data Transfer (USB). As using portable data storage devices has become a very common practice, this scenario has been developed to evaluate power consumption during a file transfer from the system hard drive to an USB Memory Device. This scenario is very similar to the previous one, exception given for the file size (which is slightly lower, near 1.8 GB) and the file destination, which is the logical drive of the USB Device.

6 - Image Browsing/Presentation. This scenario evaluates power consumption during another common usage pattern: a full-screen slide-show of medium-size images. This scenario simulates a presentation or the activity of browsing through a series of images. In this scenario, the system user opens a PDF File composed of several images, using the Acrobat Reader application. It sets the Full-Screen visualization, then manually switches through the images every 5 seconds.

7 - Skype Call (Video Disabled). As the diffusion of broadband networks increases, usage scenarios that make a more intensive use of the Internet than Web Navigation and E-Mails also become more common. For this reason, we developed the Skype scenario. Skype is the most used application for video calls and conferences among private users. For the purpose of this scenario, a dummy Skype Account was created, and the Skype application was deployed on the test machine. Then, for each run, a test call is made to another machine (which is a laptop situated in the same laboratory) for 5 minutes, which is the prefixed duration of all scenarios.

8 - Skype Call (Video Enabled). This scenario is similar to scenario 7, but the video camera is enabled during the call. This allows to evaluate the impact of the video data stream both on power consumption and on system resources.

9 - Multimedia Playback (Audio). This scenario aims to evaluate power consumption during the reproduction of an Audio file. For the purpose of this scenario, we selected a 5-minutes long mp3 file, reproduced through Windows Media Player. WMP has been selected as a reference player, as it is the default application for multimedia content in Microsoft Windows.

10 - Multimedia Playback (Video). Same as above, but in this case the subject for reproduction is a Video File, in AVI format, of the same duration.

11 - Peer-to-Peer. Peer-to-Peer applications are extremely diffused among private users. For this scenario, BitTorrent was selected as a Peer-to-Peer paradigm, because of its large diffusion and less-variant usage pattern if compared to other Peer-to-Peer systems with more complex architectures. During this scenario, the system user starts the BitTorrent client, opens a previously provided *.torrent* archive, related to an Ubuntu distribution, and starts the download, which proceeds for 5 minutes. After every execution, the partially downloaded file is deleted, in order to repeat the scenario with the same starting conditions.

In Table 2.2 all the scenarios are summarized, each with a brief description. In addition, scenarios are classified in different categories, shown in the last column, from a functional point of view. In detail:

- *Idle* (Scenario 0): it is the basis of the analysis, evaluates power consumption during the periods of inactivity of the system.
- *Network* (Scenarios 1,2,7,8,11): it represents activities that involve network subsystems and Internet.
- *Productivity* (Scenario 3): it is related to activities of personal productivity.
- *File System* (Scenarios 4,5): it concerns activities that involve storage devices and File System operations.
- *Multimedia* (Scenarios 6,9,10): it represents activities that involve audio/video peripherals and multimedia contents.

Moreover, as anticipated in the previous section, four metrics have been selected to evaluate the system usage. These metrics were measured by means of software logging (as will be explained in the *Instrumentation* section) considering the following values:

- CPU
  - CPU Time Percentage, intended as time spent by the CPU doing active work in a second
  - CPU User Time Percentage, intended as time spent by the CPU executing user instructions (i.e. applications) in a second
  - CPU Privileged Time Percentage, intended as time spent by the CPU executing system instructions (services, daemons) in a second
  - CPU Deferred Procedure Calls Percentage, intended as time spent by the CPU executing DPC in a second
  - CPU Interrupt Time Percentage, intended as time spent by the CPU serving interrupts in a second
  - CPU C1 Time Percentage, intended as time spent by the CPU in low-power (C1) State [1]

Nr.	Title	Description	Category
0	Idle	No user input, no applications running, most of OS'automated services disabled.	Idle
1	Web Navigation	Open browser, visit a web-page, operate, close browser.	Network
2	E-Mail	Open e-mail client, check e-mails, read new messages, write a short message, send, close client.	Network
3	Productivity Suite	Open word processor, write a small block of text, save, close.	Productivity
4	Data Transfer (disk)	Copy a large file from a disk position to another.	File System
5	Data Transfer (USB)	Copy a large file from an USB Device to disk.	File System
6	Presentation	Execute a full-screen slide-show of a series of medium-size images.	Multimedia
7	Skype Call (no video)	Open Skype client, execute a Skype conversation (video disabled), close Skype.	Network
8	Skype Call (video)	Open Skype client, execute a Skype conversation (video enabled), close Skype.	Network
9	Multimedia (Audio)	Open a common media player, play an Audio file, close player.	Multimedia
10	Multimedia (Video)	Open a common media player, play a Video file, close player.	Multimedia
11	Peer-to-Peer	Open a common peer-to-peer client, put a file into download queue, download for 5 minutes, close.	Network

Table 2.2: Software Usage Scenarios Overview

– CPU C2 Time Percentage, intended as time spent by the CPU in low-power (C2) State [1]

  - CPU C3 Time Percentage, intended as time spent by the CPU in low-power (C3) State [1]

- Memory
  - Memory Page Writings per second
  - Memory Page Readings per second
  - Memory Available (KiloBytes) per second
- Hard Disk
  - Physical Disk Transfers (Read/Write) per second
  - Logical Disk Transfers (Read/Write) per second
- Network
  - Network Packets per second as seen by the Network Interface Card

The dependent variable selected for the experiment is P i.e. the instant power consumption (W). Therefore, P_n is the average power consumption during Scenario $n = 1...11$.

Hypotheses Formulation

Based on the GQM Model, the Research Question can be formalized into an Hypothesis as following.

RQ 1a. Is hardware resource usage correlated with energy consumption during software execution?

H_0: $\rho(I_{CPU}, P) = \rho(I_{Memory}, P) = \rho(I_{Disk}, P) = \rho(I_{Network}, P) = 0$
H_a: $max[\rho(I_{CPU}, P), \rho(I_{Memory}, P), \rho(I_{Disk}, P), \rho(I_{Network}, P)] \neq 0$

$\rho(x, y)$ expresses the sample correlation coefficient between variables x and y.

Instrumentation

Every scenario has been executed automatically by means of a GUI Automation Software for 5 minutes, obtaining 30 runs per scenario, each composed of 300 observations (one per second) of the instant power consumption value (W).

The test machines selected are two Desktop PCs of different generations. In Table 2.3, the Hardware/Software configuration of the machines is presented. As can be seen, the difference in terms of hardware is relevant; this will allow us to

	Desktop 1 (old generation)	**Desktop 2 (new generation)**
CPU	AMD Athlon XP 1500+	Intel Core i7-2600
Memory	768 MB DDR SDRAM	4 GB DDR3 SDRAM
Display Adapter	ATI Radeon 9200 PRO 128 MB	ATI Radeon HD 5400
HDD	Maxtor DiamondMax Plus 9 80GB Hard Drive	Western Digital 1 TB
Network Adapter	NIC TX PCI 10/100 3Com EtherLink XL	Intel 82579V Gigabit Ethernet
OS	Microsoft Windows XP Professional SP3	Windows 7 Professional SP1

Table 2.3: HW/SW Configuration of the test machine

make some evaluations about how power consumption varied over the years, with the evolution of hardware architectures.

Different software and hardware tools have been used to do monitoring, measurement and test automation. The Software tool adopted is Qaliber[2] which is mainly a GUI Testing Framework, composed of a Test Developer Component, that allows a developer to write a specific test case for an application, by means of "recording" GUI commands, and a Test Builder Component, which allows to create complex usage scenarios by combining the use cases. One of the most important features of Qaliber is its possibility to log system information during scenario execution, using Microsoft's Performance Monitor Utility. By defining a specific Counter Log, adding all the variables of interest, it is possible to tell Qaliber to start Performance Monitor simultaneously with the Scenario, thus allowing a complete monitoring of all the statistics needed for this analysis.

The measurement of power consumption was done through two different devices. For the old-generation PC, YouMeter[3] device was used. This device is capable of computing Active and Reactive Power, Voltage, Current Intensity, $Cos\varphi$. The data is stored within the YouMeter's 64kB memory and can be downloaded in a text file format via Zigbee wireless connection to a Windows enabled PC or Laptop or viewed as instantaneous readings on the installed Manager software.

[2]Qaliber - GUI Testing Framework, `http://sourceforge.net/projects/qaliber/`, last visited on December 1st, 2014

[3]Youmeter - eGlue Technologies, `https://www.youmeter.it/youmeter/prodotto-applicazioni.php`, last visited on December 1st, 2014

The device drivers were slightly modified to adapt the YouMeter recording capability to this analysis' purposes, specifically to decrease the logging interval from 1 minute (which is too wide if compared to software time) to 1 second.

For the new-generation PC, WattsUp PRO ES[4] device was used. This device is capable of measuring current power consumption (Watts), power factor, line voltage and other metrics. The data is stored within the device internal memory, and then retrievable via USB interface. The sampling rate resolution is 1 second.

[4]WattsUp Pro ES, `https://www.wattsupmeters.com/secure/products.php?pn=0&wai=0&spec=2`, last visited on December 1st, 2014

Analysis Methodology

In order to extract a Power Consumption profile for each Usage Scenario, a set of descriptive statistics was derived from the experimental data. For a single scenario, a total of 30 runs were executed, each composed of 300 observations (one per second) of the power consumption value. Thus, the calculations for the descriptive statistics were made using two approaches: firstly, the average of each run is extracted, obtaining a short vector of 30 elements, which was used as the subject of our analysis. This method allowed to speed up the calculations, and because of the decreased sampling rate, the data was less variant and showed an almost regular distribution.

Afterwards, the same analysis on the full datasets was applied, which means a total of 9000 observations. Comparing the results from these two approaches, focusing on the Index of Dispersion and the variance, the variability of a single scenario can be appreciated, which was also a useful tool for validating the experiment.

First of all, data distribution must be analysed, in order to determine the appropriate testing method for each hypothesis. The data distribution analysis was conducted using the Shapiro-Wilk normality test. Since it results pointed out that the data was not normally distributed, non parametric tests were adopted, in particular the Spearman's rank correlation coefficient, also known as Spearman's ρ. This test was also chosen due to his higher robustness in presence of many outliers, a common situation when dealing with energy measurements, due to instrumentation glitches.

Since our hypothesis is non–directional, the two-sided variant of the test will be applied. We will draw conclusions from our tests based on a significance level $\alpha = 0.05$, that is we accept a 5% risk of type I error – i.e. rejecting the null hypothesis when it is actually true.

Validity evaluation

The threats of experiment validity can be classified in two categories: **internal** threats, derived from treatments and instrumentation, and **external** threats, that regard the generalization of the work.

There are three main internal threats that can affect this analysis. The first concerns the *measurement*: measurements were taken with a sampling rate of 1 second. This interval is a compromise between the power metering devices capability and the software logging service. However, it could be a wide interval if compared to software time. In addition, the two metering devices used for the analysis are different, although they have similar characteristics. This might represent a confounding factor for the differences in energy usage.

Subsequently, *network confounding factors* could arise: as several usage scenarios involving network activity and the Internet are included in our treatments, the unpredictability of the network behaviour could affect some results. Another confounding factor is represented by *OS scheduling operations*: the scheduling of user activities and system calls is out of the experiment control. This may cause some additional variability in the scenarios, especially for those that involve the File System.

In addition, the two machines on which our tests are performed are different in terms of hardware and software configuration. This is done on purpose, because we wanted to test devices which could represent common machines used in a wide variety of scenarios, for both generations. Thus, installing an old version of an operating system on a new machine or viceversa would have altered this assumption. However, this introduces another confounding factor, but still, provides useful information regarding the evolution of these systems, even if no specific research hypotheses can be verified about the comparison.

Finally, the main external threat concerns a possible *limited generalization* of the results: this is due to the fact that the experiment was conducted on only two different test machines, which is a limited sample to be representative of a whole population.

2.1.2 Results

Preliminary Data Analysis

We present in Tables 2.4 and 2.5 the following descriptive statistics about measurements for each scenario. Tables report mean (Watts), median (Watts), standard error (S.E.) on the mean, 95% confidence interval (C.I.) of the mean, sample variance, sample standard deviation (σ), variation coefficient (the standard deviation divided by the mean), index of dispersion (variance-to-mean ratio, VMR).

Power consumptions show an excursion of about 11 W for both PCs, even if the baseline is quite different (an average of 87 W in Idle scenario for the Old PC, 51 W for the New PC). Moreover, the very low variability indexes ensure that the different samples for each scenario are homogeneous.

Hypothesis Testing

The results of hypotheses testing of the research questions are exposed in this section.

First of all, Tables 2.6 and 2.7 report the results of the Data Distribution Analysis. In Tables 2.8 and 2.9 are presented the results of the correlation test using Spearman's method, with a 95% confidence interval, applied to every couple $(watt, variable)$ for each scenario. As regards Spearman's ρ significance, using

	Old-Generation PC							
	Mean	**Median**	**S.E.**	**C.I.**	**Variance**	σ	**Var.Co.**	**VMR**
0 - Idle	86.81	86.69	0.007	0.013	0.424	0.650	0.007	0.005
1 - Web	89.09	88.57	0.011	0.022	3.372	1.836	0.021	0.038
2 - E-Mail	88.03	87.11	0.024	0.047	5.195	2.279	0.026	0.059
3 - Prod	90.12	89.40	0.025	0.500	5.862	2.421	0.027	0.065
4 - Disk	94.12	97.21	0.048	0.095	21.12	4.595	0.049	0.224
5 - USB	96.41	97.10	0.024	0.046	5.047	2.246	0.023	0.052
6 - Image	91.97	91.48	0.041	0.081	15.474	3.934	0.043	0.168
7 - Skype	91.87	91.69	0.015	0.029	1.981	1.407	0.015	0.022
8 - SkypeV	95.40	95.75	0.020	0.040	3.844	1.960	0.020	0.040
9 - Audio	88.14	87.94	0.013	0.025	1.429	1.195	0.013	0.016
10 - Video	88.61	88.57	0.009	0.017	0.677	0.823	0.009	0.008
11 - P2P	88.46	88.25	0.010	0.019	0.842	0.917	0.010	0.009

Table 2.4: Scenarios Statistics Overview: Old-Generation PC

298 degrees of freedom (since 300 observations per scenario are available) the significance level of the ρ coefficient is $\beta = 0.113$. Thus, only correlations coefficients resulting higher than this value are listed.

	New-Generation PC							
	Mean	**Median**	**S.E.**	**C.I.**	**Variance**	σ	**Var.Co.**	**VMR**
0 - Idle	51.39	51.20	0.007	0.015	0.507	0.712	0.013	0.009
1 - Web	54.05	53.9	0.014	0.028	1.883	1.372	0.025	0.035
2 - E-Mail	53.40	53.40	0.011	0.021	1.123	1.059	0.019	0.021
3 - Prod	53.09	52.70	0.016	0.032	2.369	1.539	0.029	0.044
4 - Disk	60.24	62.10	0.037	0.072	12.38	3.518	0.058	0.205
5 - USB	61.29	61.90	0.023	0.046	4.901	2.214	0.036	0.080
6 - Image	52.75	52.50	0.011	0.023	1.214	1.102	0.021	0.023
7 - Skype	56.23	56.30	0.016	0.032	2.420	1.555	0.027	0.043
8 - SkypeV	62.13	62.90	0.036	0.070	11.428	3.380	0.054	0.184
9 - Audio	52.87	52.70	0.006	0.012	0.315	0.561	0.010	0.006
10 - Video	54.14	54.00	0.007	0.013	0.420	0.648	0.012	0.008
11 - P2P	54.32	54.50	0.008	0.016	0.609	0.780	0.014	0.011

Table 2.5: Scenarios Statistics Overview: New-Generation PC

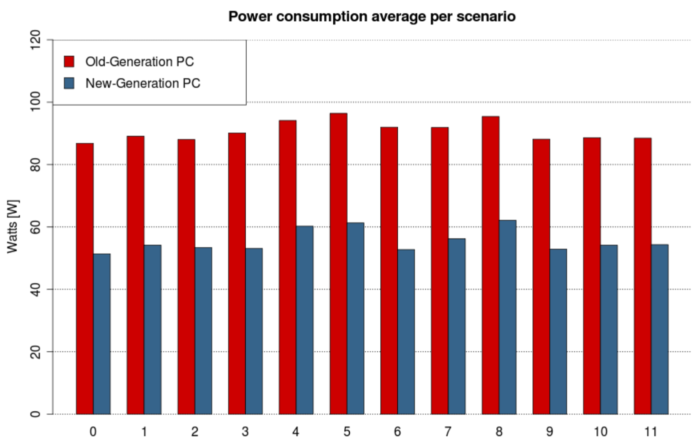


Figure 2.1: Bar Plot of per-scenario Power Consumption average values

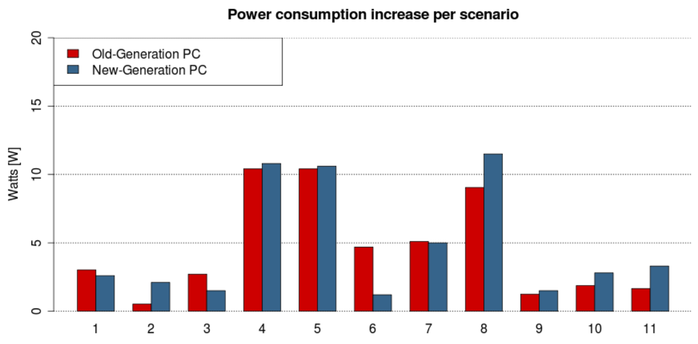


Figure 2.2: Bar Plot of per-scenario Power Consumption increase with respect to Idle

Old-Gen PC		
Scenario	**Data Distr.**	**Max p-val.**
0 - Idle	Not Normal	1.5e-63
1 - Web Navigation	Not Normal	4.4e-36
2 - E-Mail	Not Normal	9e-73
3 - Productivity Suite	Not Normal	1e-45
4 - IO Operation (Disk)	Not Normal	1.2e-46
5 - IO Operation (USB)	Not Normal	6.4e-52
6 - Image Browsing	Not Normal	1.1e-35
7 - Skype Call (No Video)	Not Normal	8.2e-30
8 - Skype Call (Video)	Not Normal	1.3e-35
9 - Multimedia Playback (Audio)	Not Normal	7.9e-54
10 - Multimedia Playback (Video)	Not Normal	1.6e-44
11 - Peer-to-Peer	Not Normal	8.9e-36

Table 2.6: Data Distribution Analysis (Old-Gen PC)

New-Gen PC		
Scenario	**Data Distr.**	**Max p-val.**
0 - Idle	Not Normal	2.2e-39
1 - Web Navigation	Not Normal	1.1e-20
2 - E-Mail	Not Normal	1.2e-19
3 - Productivity Suite	Not Normal	9.4e-29
4 - IO Operation (Disk)	Not Normal	8.7e-51
5 - IO Operation (USB)	Not Normal	2.5e-29
6 - Image Browsing	Not Normal	6.7e-22
7 - Skype Call (No Video)	Not Normal	3e-67
8 - Skype Call (Video)	Not Normal	5.2e-36
9 - Multimedia Playback (Audio)	Not Normal	5.2e-44
10 - Multimedia Playback (Video)	Not Normal	6.6e-81
11 - Peer-to-Peer	Not Normal	2.2e-35

Table 2.7: Data Distribution Analysis (New-Gen PC)

	Old-Generation PC			
Scenario Title	**Variable**	**p-value**	ρ	**R2**
2 - E-Mail	CPUC1Time.	< 0.0001	-0.36	13 %
4 - IO Operation (Disk)	CPUTime.	< 0.0001	0.35	12 %
4 - IO Operation (Disk)	CPUC1Time.	< 0.0001	-0.35	12 %
5 - IO Operation (USB)	CPUTime.	< 0.0001	0.47	22 %
5 - IO Operation (USB)	CPUC1Time.	< 0.0001	-0.47	22 %
7 - Skype Call (No Video)	CPUC1Time.	< 0.0001	-0.39	15 %
8 - Skype Call (Video)	CPUTime.	< 0.0001	0.63	40 %
8 - Skype Call (Video)	CPUUserTime.	< 0.0001	0.53	28 %
8 - Skype Call (Video)	CPUC1Time.	< 0.0001	-0.7	49 %
11 - Peer-to-Peer	MemoryKByteAvailable	< 0.0001	-0.34	12 %

Table 2.8: Spearman's ρ Coefficient between Power and Resource variables

	New-Generation PC			
Scenario Title	**Variable**	**p-value**	ρ	**R2**
2 - E-Mail	CPUUserTime.	< 0.0001	0.42	17 %
2 - E-Mail	CPUPrivTime.	< 0.0001	0.43	18 %
3 - Productivity Suite	CPUUserTime.	< 0.0001	0.33	11 %
4 - IO Operation (Disk)	PhysicalDiskTransfers	< 0.0001	0.45	20 %
4 - IO Operation (Disk)	LogicalDiskTransfers	< 0.0001	0.45	20 %
4 - IO Operation (Disk)	MemoryPages	< 0.0001	0.44	19 %
4 - IO Operation (Disk)	MemoryKByteAvailable	< 0.0001	-0.54	29 %
4 - IO Operation (Disk)	CPUC3Time.	< 0.0001	-0.59	35 %
4 - IO Operation (Disk)	CPUTime.	< 0.0001	0.55	31 %
4 - IO Operation (Disk)	CPUUserTime.	< 0.0001	0.58	34 %
4 - IO Operation (Disk)	CPUPrivTime.	< 0.0001	0.39	15 %
6 - Image Browsing	CPUUserTime.	< 0.0001	0.34	12 %
7 - Skype Call (no video)	NetworkPkts	< 0.0001	0.62	39 %
7 - Skype Call (no video)	MemoryKByteAvailable	< 0.0001	-0.45	20 %
7 - Skype Call (no video)	CPUC3Time.	< 0.0001	-0.66	43 %
7 - Skype Call (no video)	CPUTime.	< 0.0001	0.52	27 %
7 - Skype Call (no video)	CPUUserTime.	< 0.0001	0.63	39 %
8 - Skype Call (Video)	NetworkPkts	< 0.0001	0.67	46 %
8 - Skype Call (Video)	MemoryKByteAvailable	< 0.0001	-0.62	39 %
8 - Skype Call (Video)	CPUC3Time.	< 0.0001	-0.88	77 %
8 - Skype Call (Video)	CPUTime.	< 0.0001	0.87	76 %
8 - Skype Call (Video)	CPUUserTime.	< 0.0001	0.9	81 %
9 - Multimedia (Audio)	MemoryKByteAvailable	< 0.0001	-0.34	12 %
11 - Peer-to-peer	NetworkPkts	< 0.0001	0.45	20 %
11 - Peer-to-peer	MemoryKByteAvailable	< 0.0001	-0.42	18 %
11 - Peer-to-peer	CPUPrivTime.	< 0.0001	0.35	12 %

Table 2.9: Spearman's ρ Coefficient between Power and Resource variables

2.1.3 Discussion

The collected data shows several facts. As observed in Figure 2.3, in both our test machines every usage scenario consumes more power than the Idle scenario. This difference is even more evident in the New-Generation PC, where we witness a power consumption increase up to 20%.

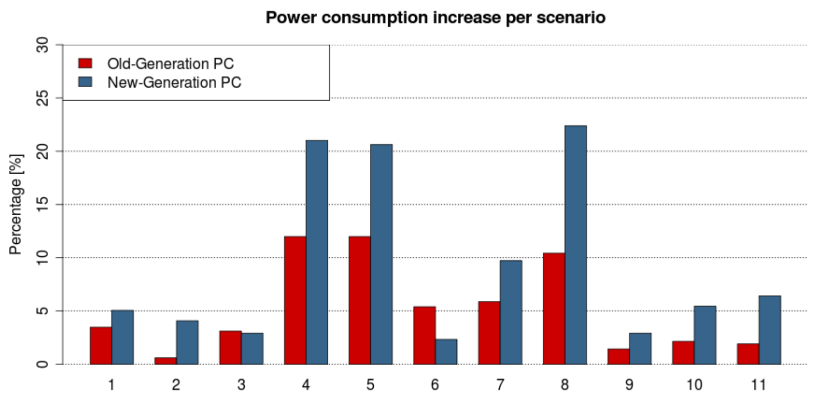


Figure 2.3: Per-scenario Power Consumption increase with respect to Idle (in percentage)

As can be seen from Tables 2.4, 2.5 and Figure 2.1, the most power-consuming scenarios are those that involve File System, followed by Skype (both with and without Video Enabled) and Image Browsing. From the hardware point of view, these scenarios are also the most intensive in terms of system resources. This also implies that resource utilization can be an accurate way to estimate their power consumption. For instance, the power consumption profile of Skype is very different (about 4-5 Watts in average) with and without enabling the Video Camera.

Another interesting question that arises from the analysis is, in case of applying these Scenarios in groups, if their power consumption would follow a linear composition rule (thus summing up the values). That is, for example, supposing a composed Usage Scenario S that involves a Skype Call, a Web Navigation and a Disk Operation performed simultaneously, their linear composition would give, on our Old-Gen PC, an estimated Power Consumption per second of

$$P_{idle} + \Delta P_S = 86.81W + 21.33W = 108.14W$$

introducing a 25% overhead on power consumption. On the New-Gen PC, the estimated Power Consumption would be

$$P_{idle} + \Delta P_S = 51.39W + 24.90W = 76.29W$$

which gives a 48% overhead on power consumption.

Taking a look at the results of the correlation analysis, we can observe that the coefficients related to the New-Gen PC are higher than those of the Old-Gen PC. This may suggest that as hardware evolves, the software usage is even more significant for determining the power consumption of the system. This assumption is confirmed by Figure 2.3, where we can observe that the percentage increase of power consumption for the New-Gen PC is higher, in most cases, than the Old-Gen.

However, it is remarkable that, for both machines, the variables that show higher correlation coefficients are undoubtedly those related to CPU usage and memory usage, namely CPU Total Time, CPU User Time, Memory Available and Memory Pages. High coefficients are also present in the Hard Disk Index, but only in those scenarios that, unsurprisingly, involve File System operations. This means that CPU and memory have a greater influence upon power consumption related to the others selected for the analysis.

Another observation is that, as expected, power consumption has always a *negative* correlation with the time spent by the CPU in the low-power C1 and C3 states and with the available memory. This suggests that using more memory has a positive correlation with power, as expected. This is also a confirmation that the analysis was conducted with the right premises.

Moreover, as expected, the scenarios who exhibit higher correlations are those who use more resources, such as Skype and IO scenarios. In particular, the Skype scenario with video enabled has a strong correlation with the CPU usage, probably because the real-time video elaboration makes the CPU the dominant resource for power consumption.

2.2 Software Energy Measurement and Modeling: State-of-the-art

From our experimental results, it clearly emerges that the usage patterns of IT resources by software applications have a clear correlation with energy consumption. This suggests that resource usage can be used as a *predictor* for software energy consumption. RQ 1b focuses on this aspect, namely:

RQ 1b. How can software properties be used as a predictor for hardware energy consumption?

In general, in order to predict software energy consumption, we need to:

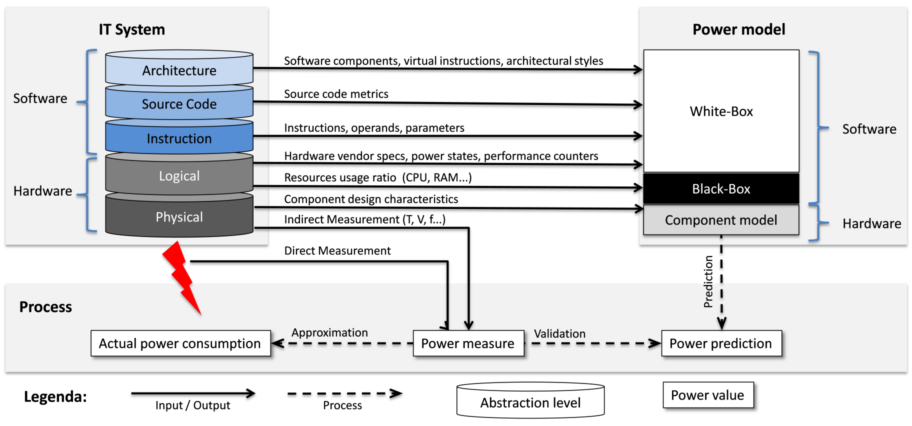


Figure 2.4: Overview of measurement, prediction and modeling approaches of software energy consumption

- measure the energy consumption of IT devices to derive quantifiable relationships between resource usage and energy consumption;
- develop a general software energy model that embeds those relationships and takes resource usage data as input, providing an energy consumption prediction.

To answer RQ 1b, we surveyed the existing literature looking for the most used solutions for software energy measurement and modeling. We classified the different solutions we found in a conceptual overview provided in Figure 2.4: on the left side of the picture, you see an IT system represented in terms of abstraction layer. Out of each layer, it is possible to extract relevant data that concerns software and energy consumption. On the right side, you see our classification of the existing prediction models for software energy consumption. Depending on whether internal properties of software are used for prediction, models can be either *white-box* or *black-box*. The accuracy of the prediction can be validated and improved by using measurements (either *direct* or *indirect*) of energy consumption. In the remainder of this section we describe the measurement and modeling solutions in detail.

2.2.1 Software Energy Measurement

In order to empirically characterize the relationship between the resource usage and energy consumption, we need to be able to measure it with sufficient precision. However, this is a hard task, mainly due to the complex relationships between software systems and the hardware configuration of their execution environment [126].

Many techniques have been developed for energy consumption measurement of IT systems. They can be classified in direct or indirect [47]. A *direct* measurement occurs when energy consumption is measured directly, right on the device(s), through an external meter [34]. Direct measurements are typically performed on self-contained systems, i.e. PCs [47] or embedded devices [104, 126]. As an example, the experiment presented in the previous Section was conducted making use of a direct measurement approach, as we obtained our energy consumption data by plugging our test machines to an external power meter.

Indirect measurements, instead, derive energy consumption from secondary quantities (e.g., voltages, currents, temperatures). These quantities are usually observed by means of sensors embedded on the device and made available via software (software probes).

JouleUnit [134] is an example of a framework for energy profiling of software applications using both direct and indirect measurements. It provides a general software-testing environment, able to interface itself with both hardware- and software-based profilers (i.e. power metering devices for direct measurements), or software probes provided by the device architecture for indirect measurements. While direct measurements are, in general, more accurate, they introduce several overheads: cost overhead due to metering equipment, an energy consumption overhead because metering equipment has to be powered, and a skill overhead because in some cases, installing and using this equipment requires a certain amount of technical knowledge [146]. Indirect measurements are in general less expensive, because they make use of already available information from the hardware layer. However, their accuracy depends on how much information the hardware layer (i.e. hardware vendors) provides. Often, hardware vendors do not provide access to all sensors installed on their products. In addition, when many software layers are built on top of hardware (as in complex architectures like Cloud Computing environments), tracing indirect measurements to software behavior becomes extremely challenging.

2.2.2 Energy Modeling

Another option to assess software energy efficiency is using software-based energy consumption prediction. A prediction is defined as the process of determining the magnitude of a variable at some future point of time [29]. Unlike measurements, these methods do not directly observe physical quantities, but rather make use of an energy model [119], typically characterized through empirical methods.

A fundamental distinction must be made between hardware and software energy models. Hardware energy models simulate the behavior of hardware components through circuit-level analysis and subsequently estimate the overall energy consumption through composition of the different logical blocks [104]. However, those models typically miss system-level effects, such as those related to temperature. Moreover, they are mostly focused on modeling processor units [146, 102, 105] because CPUs are considered as the primary source of energy consumption in a computer system.

Software energy models express energy consumption as a function of software-related metrics [104] that act as predictors, i.e. independent variables that have a relationship of some sort (linear, non-linear) with energy consumption. Those models differ significantly depending on the system under test, experimental setting, adopted parameters, abstraction level, etc.

An example of a (linear) model underlying the energy consumption of a generic IT device can be represented as:

$$E_t = E_i + \sum_{c \in Components} E_{Hc} \cdot Sw_c \text{ where } \quad 0 \leq Sw_c \leq 1 \tag{2.1}$$

The total energy consumption E_t of an IT device – when turned on – is composed by an E_i part that is present even when the device is sitting idle. The additional consumption depends on the individual hardware components maximum consumption E_{Hc} which is modulated by how much work the software demands them to do, Sw_c. Depending on the software requests the hardware component may run at full throttle or remain idle.

Choosing the most meaningful metrics as predictors, isolating the platform-dependent aspects from the software abstractions, eliciting the appropriate software constructs to analyze are only some of the research problems that must still be addressed. The accuracy of a predictive model, of course, strongly depends on the chosen predictors. Usually a preliminary correlation analysis is done, in order to extract the most meaningful predictors to be embedded in the model. As with other predictive modelling techniques, accuracy improvements can be achieved via a larger training dataset, or including other factors – taking into account the risks of overfitting.

There are two classes of software energy models, that we name *white-box* and *black-box*. *White-box* models aim at estimating software energy consumption from internal properties of the software under test. Depending on the abstraction level (or granularity), we can distinguish between instruction-level, function-level or block-level models [104]. In white-box software energy models, *performance counters* are commonly used [104]. These counters are embedded into hardware components like CPUs or memory banks and made available through software. They record component-level events and signals, through which a programmer can associate the execution of specific instructions with hardware states. In this way, it is possible to model the energy consumption of software tasks. A first example of such a model was provided, for Assembly language, by Tiwari et al. [126] in 1994. A more recent example is eProf [111], a profiler able to relate energy consumption to code locations using a probabilistic sampling of hardware performance counters. An example of instruction-level model is provided by Song et al. [121]. They model the energy of a single instruction as a composition of static energy, independent from software activity and proportional to execution time, and dynamic energy, as a function of workload, architectural and physical parameters. Performance counters can also be used to estimate the energy consumption of virtual processors, such as the energy-aware virtual scheduler developed by Kim et al. [62]. Along with performance counters, another common approach is the usage of power states, particular configurations of hardware components that operate at different levels of performance and energy consumption [145]. Since it is possible to monitor the current state of a component via software, this information can be used as an input to a energy model. For example, SEProf [128] and eLens (for mobile devices) [48] are two energy-profiling tools that associate source code constructs (either at block, method or instruction level) to an energy consumption value of one or more system components. All approaches above share an open research problem: hardware specifications are not always available for all platforms, and for this reason white-box models are usually platform-dependent. To be able to generalize a white-box model means to elicit software properties that impact energy efficiency and that are independent from the deployment platform.

A complementary approach, that evaluates software energy consumption from the execution phase, uses black-box models. *Black-box* models consider software as a self-contained entity, without digging into its internal structure [88]. These models express the relationship between runtime metrics, typically usage ratios of system resources (CPU, RAM, etc.) and energy consumption. This relationship is usually modeled through linear regression, or other statistical inference techniques. A first, explorative attempt was performed by Sinha [119] that built a model for energy consumption on a StrongARM processor using as input the number of CPU cycles needed by an application. However, the model was strongly tailored upon that specific CPU. A more recent example is provided by Morelli et al. [88] who presented a compositional model that relates the resource usage data of different applications with energy consumption benchmarks. With this data, they compose a measurement matrix, which is then fed into a linear algebra model for energy consumption prediction. Another example of a black-box, resource-based model for mobile devices is presented by Palit et al. [93]. Their results show how, in mobile contexts, the impact of CPU and other components such as wireless devices, may be very significant depending on the considered scenario. In mobile devices, other sources of information may be available for black-box analysis: for example, battery discharge patterns. Carat [92] analyzes these patterns by comparing them with an *a priori* probability distribution, in order to detect abnormal energy consumption by software applications. In cloud environments, the black-box approach is adopted at cloud node/resource level (e.g. for task workload [23], energy management and performance tradeoffs [144], infrastructure management [117]). Most black-box models focus on CPU energy modeling. In fact, CPU is certainly one of the most energy-consuming components. Moreover, its behavior is highly influenced by software. Technological advancements resulted in highly flexible processor units, capable of several operational modes with different energy consumption profiles. However, in some cases, for example in mobile devices [93], certain components, like GPS modules and Bluetooth/WiFi antennas, may significantly impact energy consumption. This implies that modeling the CPU is not enough and black-box models should define predictors for other components, depending on the context.

2.3 Conclusion

In this chapter, we provided a twofold contribution: an empirical experiment on desktop computer systems and a literature survey on software power models.

The experiment assessed quantitatively the energetic impact of software usage. It consisted in building up common application usage scenarios (e.g.: Skype call, Web Navigation, Word writing) and executing them independently to collect power consumption data. Each single scenario introduced an overhead on power consumption, which may raise up to 20% for recent systems: if their power consumption would follow a linear composition rule, the impact could be even higher.

The relationship between usage and power consumption was also analysed in terms of correlation between resource usage. From our results, we can safely reject the null hypothesis, as it clearly stands:

$$H_a : max[\rho(I_{CPU}, P), \rho(I_{Memory}, P), \rho(I_{Disk}, P), \rho(I_{Network}, P)] \neq 0$$

Although a clear linear relationship did not arise, the analysis showed that some resources drive power consumption more than others, such as memory and CPU usage. This gives the answer to RQ 1a, namely "*Is hardware resource usage correlated with energy consumption during software execution?*". Our experiment also gives us the indication that modern systems, although being more energy efficient in standby and idle states, due to their higher scalability, are even more sensible to the energy consumption impact of software usage.

Our experimental results suggest that using resource usage data, it could be possible to predict software energy consumption. To assess the feasibility of such an approach, we surveyed the state-of-the-art in assessing software energy efficiency and framed it in Figure 2.4. Our analysis concludes that many tools and techniques for software power modeling and measurement are already widely available. This answer RQ 1b, namely "How can software properties be used as a predictor for hardware energy consumption?" We showed that resource usage data is a reliable predictor for energy consumption. In summary, the whole chapter provides an answer to RQ1, namely '*What is the correlation between software properties and hardware energy consumption?*": the software impact over hardware energy consumption is significant, it is driven and characterized by the resource usage ratio and as such it can be predicted by them.

By now, the reader should have a clearer picture of the key contribution of software in the energy consumption performed by hardware. Moreover, in this chapter we showed the potential of the usage of power models for energy-aware applications: embedding power prediction models in the application logic enables it to alter its behavior according to the energy status of the environment. For example, in mobile contexts (characterized by battery constraints) energy awareness is a crucial requirement. A common energy-driven policy regards offloading to the cloud [33], i.e. deciding whether to perform a task locally or delegating it to a cloud computing infrastructure. This is a non-trivial problem, because e.g. the energy spent for transferring data to the cloud could be higher than the savings achieved through offloading the computation [93]. These types of tradeoffs also appear in other contexts: e.g. data compression is usually regarded as a way to reduce energy consumption by decreasing I/O activity. However, it has been shown [111][93] that in some cases the computational effort for compression/decompression operations wastes significantly more energy than the amount saved by performing less I/O operations.

This leads to the need for knowledge in software energy efficiency: reusable software practices, at the level of architecture and design, aimed at capturing these tradeoffs for improving software energy efficiency. These practices have to be validated, in order to avoid the second-order effects presented before. In the following chapter, we describe an empirical experiment aimed at evaluating the impact of two best practices for energy-efficient software.

3 Empirical Evaluation of Best Practices for Energy-Efficient Software Development

Current state-of-the-art does not provide empirically validated guidelines for developing energy efficient software. In this chapter, we present the design and results of an empirical experiment to assess the impact of two best practices for energy efficient software development, hence answering RQ 2. We elicited the practices from previous publications in academic and industrial literature. This chapter also aims at identifying the possible trade-offs between energy consumption and other software properties. We performed an empirical experiment in a controlled environment, where we applied two different best practices on two software applications. We then performed a comparison of the energy consumption at system-level and at resource-level, before and after applying the practice. Our results show that both practices are effective in improving software energy efficiency, up to a 25% improvement. We observe that after applying the practices, resource usage is more energy-proportional. We also provide our reflections on empirical experimentation in software energy efficiency. Our contribution shows that significant improvements can be gained by applying best practices during design and development.

3.1 Introduction

As showed in the previous chapter, many researchers have been working on sophisticated software power models [119, 58], able to estimate and predict the energy consumption of software applications through different parameters. However, this effort has not been translated yet into reusable information for practitioners and developers to create energy-efficient software applications. A step in this direction has been made by Larsson et al. [77] from Intel Corp., which provided a number of guidelines and best practices for creating energy-efficient software.

However, little to no validation has been performed on those practices, and their effectiveness in terms of energy consumption has not been precisely quantified.

To understand how software can impact on energy consumption on the large scale, consider the following example[1]: after launch, the popular Youtube video of the "Gangnam Style" song reached a record amount of visualizations during the first year after its publication – roughly 1.7 billion. The amount of energy used by Google to transfer 1MB across the Internet (as reported by the company on their website[2]) is 0.01kWh (a rough average), and displaying it uses 0.002kWh (depending on the destination device). Hence, the energy needed to stream and display the "Gangnam Style“ video is 0.19 kWh. Multiplying this amount of energy by the 1.7 billion visualizations gives 312 GWh of total energy consumption, which is roughly the yearly energy demand of a city of 22.000 inhabitants (as an example, the city of Isernia, Italy, consumed 340 GWh of electricity in 2013 [125]).

This impressive amount of energy may hide huge wastes. A complex software architecture lies behind modern web applications and services (e.g. webservers, database servers, middleware) and countless instances are executed every second in physical and virtual environments. Even a tiny optimization on a single software application, on such a massive scale, could potentially lead to significant energy savings. For this reason, software architects and developers need to think about energy efficiency and a solid knowledge base is needed to provide guidance in building energy-efficient software.

The aim of this chapter is assessing the impact of best practices for energy-efficient software development on energy consumption. This chapter answers RQ 2, namely:

RQ 2. What is the impact of using best practices for software energy efficiency?

Our work follows the guidelines for empirical experimentation in software engineering provided by Wohlin et al. [138] and Basili et al. [11]. For the purpose of this experimentation, we applied two practices in well-known open source software applications (the Apache WebServer and the MySQL Database Server) that will serve as test cases. These applications were executed in a controlled environment (the Software Energy Footprint Lab, SEFLab [34]). During the experiment, we gathered two types of data: power consumption (both of the execution environment as a whole and of the single hardware components) and resource usage of the different hardware components. Then, we performed hypothesis testing on the data to answer our research questions. Besides assessing the energy impact of each practice, we elicited, for each test case implementation, the software metrics

[1]`https://www.2degreesnetwork.com/groups/energy-carbon-management/resources/gangam-style-it-sustainable/`

[2]`http://www.google.com/green/bigpicture/`

and factors that we identified as most relevant for energy consumption purposes. From this comparison, we extracted meaningful information to further define the complex relationship between software and energy.

This chapter is organized as follows: Section 3.2 presents an overview of previous empirical studies on the energy consumption of software applications. In Section 3.3 we present our study design, in terms of subjects, objects, dependent/independent variables and instrumentation. In Section 3.4 we describe how the experiment was executed. In Section 3.5 we discuss the validity aspects and possible threats arising from our experiment design and execution. In Section 3.6 we present our experimental results for each practice and hypothesis testing. In Section 3.7 we answer our research questions and discuss the implications of our findings. In Section 3.8 we draw conclusions and outline our future research efforts.

3.2 Related Work

A number of empirical experiments on software energy consumption has been conducted as software energy efficiency became a popular research topic. In this section, we present those which are more related to our contribution, ordered by publication date, and summarize their findings. In Table 3.1 we give a more structured overview: we list the *purpose* of the study, the experimental *context* (e.g. on-line vs. off-line [138]), the *subjects* selected for the study and the *testbed* on which the energy measurements were performed (where applicable). As criteria for selection, we focused on the viewpoint of developers: hence, we selected studies analyzing the impact of programming techniques or practices on energy consumption, as well as studies that try to empirically characterize energy-intensive code elements.

1. Capra et al. [18] analyze the impact of application development environments over the energy efficiency of software applications. They propose a measure of the impact of application environments on the development process, called *framework entropy*, and evaluate it over a set of 63 open source applications. Hereby we list the main findings of this work:

    - *Finding 1.* A high framework entropy is beneficial for the energy efficiency of small and medium applications.
    - *Finding 2.* A high framework entropy is detrimental for the energy efficiency of large applications.
    - *Finding 3.* Different functional types of applications have different energy efficiency levels.

Table 3.1: Summary of the related work.

Ref.	Purpose	Context	Subjects	Testbed
[18]	Evaluate the energy impact	Off-line, single-object	Application Development Environments (63 open-source projects)	Server
[107]	Evaluate the energy impact	Off-line, single-object	Software Design Patterns (15 Design Patterns in 3 categories)	Embedded System
[89]	Evaluate the energy impact	Off-line, multi-object variation	Algorithms and Programming Languages (8 Towers of Hanoi implementations)	Server
[52]	Trace the evolution of software energy consumption	Off-line, multi-object variation	3 products (Firefox, Vuze, rTorrent) in different versions and scenarios	Laptop PC
[69]	Evaluate the energy impact	Off-line, multi-object variation	8 Distributed Programming Abstractions on 5 scenarios	Server-Client
[97]	Evaluate the energy impact	Off-line, multi-object variation	3 Thread Management Constructs on 8 different benchmarks	Server
[78]	Evaluate the energy impact	Off-line, single-object study	3 best practices for energy-efficient programming in Android	Smart-phone
[79]	Identify most energy-greedy API calls	Off-line, case study	55 Android applications	Smart-phone
[98]	Identify most used programming solutions for energy-efficient software	On-line, thematic analysis	325 questions and 558 answers on Stack Overflow about energy-efficient software	N/A

- *Finding 4.* ERPs, text, image editors and games are less energy efficient than FTP clients and servers, and calendars.

2. Sahin et al. [107] investigate the energy impact of using software design patterns. They consider a set of 15 design patters and evaluate the energy consumption of a "proxy" application developed on purpose for the study. The application is evaluated in two versions, before and after applying the design pattern. Hereby we list the main findings of this work:

- *Finding 1.* The impact of applying a design pattern varies greatly, from less than 1% to more than 700%, among the considered patterns.
- *Finding 2.* The impact of design patterns is not consistent with respect to the pattern category (i.e. Creational, Structural, Behavioral [38]).
- *Finding 3.* The impact of design patterns cannot be predicted by looking at how it influences high-level design artifacts.

3. Noureddine et al. [89] analyze the energy impact of programming languages and algorithmic choices. The impact is evaluated through a low-level library called *PowerAPI* on 8 different implementations of the Towers of Hanoi program, varying the implementation language and the used algorithm (recursive vs. iterative). Hereby we list the main findings of this work:

   - *Finding 1.* The algorithm choice has a significant impact on energy consumption. The recursive algorithm is more energy-efficient than the iterative one.
   - *Finding 2.* The chosen programming language has a significant impact on energy consumption as well. The Java implementation is more energy efficient than the others, not considering compiler optimizations.
   - *Finding 3.* The impact of compiler optimizations is also relevant. Compiling the C++ implementation with the O2 compiler option increases energy efficiency significantly.

4. Hindle [52] investigates the impact of software change on power consumption, and the relationship with software metrics. Subjects are 3 applications: Firefox, Vuze, rTorrent. For each application a set of different versions and releases is selected. Hereby we list the main findings of this work:

   - *Finding 1.* Power consumption is not consistent among different versions.
   - *Finding 2.* Performance evolutions can affect power consumption in multiple ways.
   - *Finding 3.* No significant correlation was found between static OO-related software metrics (e.g. coupling, cohesion, fan-in/fan-out) and power consumption. Process-related metrics (e.g. added/removed lines, file churn) exhibit positive correlation with power consumption in a limited amount of cases.

5. Kwon and Tilevich [69] analyzed and evaluated the impact in terms of energy consumption of major Distributed Programming Abstractions (DPA) when developing communication mechanisms for mobile devices, such as RPC, RMI

or SOAP. Authors implemented 8 versions of different benchmarks for middleware platforms, each version adopting a specific communication abstraction. Hereby we list the main findings of this work:

- *Finding 1.* Binary-based DPAs (eg. raw sockets) are more energy efficient than XML-based ones, because of the smaller overhead in communication data.
- *Finding 2.* Asynchronous DPA mechanisms have no additional energy costs.
- *Finding 3.* Marshaling/unmarshaling consume more energy on network communication than on CPU processing. Serialization protocols are more energy-efficient in high-throughput networks.

6. Pinto et al. [97] analyzed and evaluated the impact in terms of energy consumption of 3 different thread management strategies, i.e. *explicit threading, thread pooling, work stealing*, applied on 8 different benchmarks. They also analyzed how energy consumption varies in relationship to the number of active threads. Hereby we list the main findings of this work:

   - *Finding 1.* Different thread management constructs have different impacts on energy consumption. For I/O-bound programs, *explicit threading* is the most energy-efficient, whereas *work stealing* is the least. For highly parallel benchmarks, the opposite holds.
   - *Finding 2.* Energy consumption typically increases as the number of threads increases, and then gradually decreases as the number of threads approaches the number of CPU cores.
   - *Finding 3.* Being faster is not synonymous of being greener. Sequential execution often leads to the least energy consumption, whereas parallel execution leads to improved energy/performance trade-off.

7. Li and Halfond [78] evaluate the impact of 3 best practices for Android application development extracted from the official Android developers community forum. The practices can be summarized as: bundle small HTTP requests, reduce memory usage and improve performance to decrease energy consumption. Authors developed three small software applications to test the impact of each practice. Hereby we list the main findings of this work:

   - *Finding 1.* Bundling small HTTP requests could save energy.
   - *Finding 2.* Higher memory usage only slightly increases the average energy consumption of each access.
   - *Finding 3.* Avoiding references to the array length for each loop iteration can save energy.

- *Finding 4.* Directly accessing fields instead of accessing them through methods can save energy. The reason is that the virtual methods that are used to access field values are expensive operations.
- *Finding 5.* Static invocation appears to be more energy-efficient in Android.

8. Linares-Vásquez et al. [79] aim at identifying whether some API calls are more energy-consuming than others, and if sequences of API calls (patterns) repeat themselves frequently, causing anomalies in energy consumption. The study analyzed the execution traces of 55 Android applications, looking for the most energy-greedy Android API calls. Hereby we list the main findings of this work:

   - *Finding 1.* APIs related to *GUI & Image Manipulation* and *Database* are the most energy-consuming.
   - *Finding 2.* Using getters and setters when accessing internal class fields causes high energy consumption. This finding is coherent with the previous study, and creates a trade-off between information hiding and energy efficiency.
   - *Finding 3.* Refreshing application views and widgets causes high energy consumption.

9. Pinto et al. [98] mined the StackOverflow platform to find the most common problems regarding energy efficiency, their causes, and the most recommended programming solutions for energy-efficient software. The study found a total of 325 questions and 558 answers from more than 800 software developers. Hereby we list the main findings of this work:

   - *Finding 1.* There are misconceptions about software energy consumption, like confusion between power and energy and the correlation between energy and performance.
   - *Finding 2.* The major causes for energy consumption according to developers are: unnecessary resource usage, hidden background activities, excessive synchronization.
   - *Finding 3.* Among the most suggested solutions, those matching with the scientific state-of-the-art were: reduce I/O to a minimum, buffer I/O commands, avoid polling, use efficient data structures.

Although our literature search was not conducted systematically, we performed an extensive review that allows us to make some considerations. As emerges from those findings, there are many preliminary insights and hypotheses about software energy efficiency. For example, it seems that large applications with many subsequent versions tend to be less energy-efficient than smaller ones. However, we also observe a certain degree of conflict and uncertainty. For example, some studies seem to show that high-level abstractions and languages are less energy-efficient than low-level ones, while others conclude the opposite. Obviously, the studies analyze different entities (i.e. populations). Many researchers focus on mobile applications (in mobile environments, battery life is obviously a high priority), other focus on specific application domains (e.g. Information Systems) or technologies (DPAs). However, the studies also differ in terms of research approach: each study adopts a different instrumentation and study design. We claim that such a difformity of approaches prevents practitioners from having sound reference and guidance when building energy-efficient software.

3.3 Experiment Planning

This section describes our experiment planning, in terms of dependent and independent variables, hypotheses formulation, and instrumentation [138]. A summary is provided in Table 3.2.

Table 3.2: Summary of experiment planning phase.

Object of Study	Best Practices for software energy efficiency
Subjects	Open-source software applications
Independent variable	Application workload
Dependent variables	– Energy consumption values (at system- and resource-level)
	– Resource usage measures
	– Software execution measures

3.3.1 Variable Selection

The main **objects** of our study are best practices for software energy efficiency. We elicited those practices inspired by academic literature and industry [118, 110, 56, 122, 5, 77, 46] and collected them in a wiki[3] to share them with academics and practitioners.

For the purpose of this evaluation, we selected two practices from our wiki: *Use efficient queries* and *Put application to sleep*. Those practices were selected for two main reasons: the high relevance for practitioners, as they can be applied

[3]`https://wiki.cs.vu.nl/green_software/index.php/Main_Page`

in a wide context of software applications, and ease of implementation. The practices are described in Tables 3.3 and 3.4, respectively, using the template described in [46]. More details and the rationale behind our implementation choices will be presented in Section 3.4.

Due to the nature of the practices and their formalization, it was not possible to automate and randomize their application to the subjects, i.e. software applications. Hence, our empirical study qualifies as a **quasi-experiment**, or more appropriately as a **single-case mechanism experiment** [133], where we test the cause-effect behavior of the best practices on selected experimental subjects. Our choice fell upon two commonly-used, open-source products: the Apache Web Server and the MySQL Database Server. The same criteria used for object selection guided this choice: the wide usage of these products ensures relevance for practitioners, and their open-source nature allowed us to easily access their source code for instrumentation purposes.

Our study population is the set of all possible executions of these two software applications in two different scenarios i.e. with and without the practice. Out of our population, we draw a sample of 10 executions per application per scenario, thus a total of 40 executions.

Table 3.3: Description of Practice 1: Use efficient queries.

Logical name	Use of efficient queries
Category	Databases
Description	Most of Web applications of any size involve the use of a database. Typically, a Web application allows the addition or creation of new records (for example, when a new user registers on the site), and the reading and searching operations of many records in a database. Consequently, the traditional performance bottleneck of Web applications comes from the database. It is often caused by reading operations of a large number of records, or reading operations whose complexity requires an expensive data processing time by the database.
Rationale	Often, database queries perform complex operations, such as ordering or indexing. Those operations are done to increase the application performance at the expense of energy efficiency. Hence, limiting the utilization of indexation mechanisms or unnecessary ordering operations (use of ORDER BY keywords) can mitigate the energy consumption of our queries.
Source	Green Software Wiki
Keywords	database, coding, query

The **dependent variables** we monitored and analyzed for answering our

Table 3.4: Description of Practice 2: Put application to sleep.

Logical name	Put application to sleep
Category	Energy-efficient coding
Description	This practice makes use of a sleep function (or equivalent) that puts a process (or main thread) in sleep mode for a specific period of time, i.e. enter the Not Runnable state. This programming technique enables to suspend a thread or process, and thus its use of CPU resources, while continues executing other threads or processes until the *sleep mode* has finished. Once the sleep mode is over the thread or process is allowed to continue making use of CPU resources.
Rationale	A proper use of the Sleep function (e.g. when the application is no longer active, waiting for I/O or other signals) allows to reduce CPU utilization, and consequently improves the energy efficiency of an application.
Source	Green Software Wiki, Wikipedia
Keywords	sleep, coding, thread

RQs are: the energy consumption at system-level to assess the energy impact of the practices; the energy consumption values at resource-level and its usage ratio to identify the most affected resources; and software execution measures (response time, number of request/query served) to determine their relationship with energy consumption and how the application of the practice affects them.

The **independent variable** for our experimentation is the application workload, i.e. the parameters we used for benchmarking (e.g. total number of requests, database size).

3.3.2 Hypotheses Formulation

In the following we formulate the hypothesis that guides our experimentation, starting from our research question.

RQ 2: What is the impact of using best practices for software energy efficiency?

The impact (ΔE) is measured in watt-hours (Wh) and expresses the difference between the energy consumption at system level before (E_0) and after (E_1) applying the practice, namely:

$$\Delta E = E_1 - E_0$$

Thus, we have:

$H2_0$: $\Delta E \approx 0$
$H2_a$: $|\Delta E| \gg 0$

The null hypothesis implies a negligible impact of the practice over energy consumption. The alternative hypothesis represents instead an evident and significant impact of the practice.

3.3.3 Instrumentation and Testbed

In the following we describe the instrumentation we used in our experimentation, in terms of hardware and software tools. The hardware tools were provided by the Software Energy Footprint Lab (SEFLab) [34], which served as our laboratory environment.

Hardware

The test machine is a Dell PowerEdge SC1425 server, with the following specifications:

- 2x Intel Xeon CPUs, 3.2GHz
- 4x Infineon 1GB DDR2-333 SDRAM
- Intel E7520 chipset
- 1x Maxtor 7L250S0 250GB SATA150 HDD
- Dell power Supply Unit 450W

The instrumentation on this server consists of two Texas Instruments Data Acquisition Boards (DAQs) connected to the power supply channels of the single resources (e.g. CPUs, memory banks), in order to record the power consumption data of the different components of the server. In addition, this server is also equipped with a Wattsup PRO[4] meter to record system-level power consumption.

Software

The Software instrumentation consists of tools able to collect power consumption data and software-related measures, along with resource usage information. All this data is timestamped, synchronized and stored in comma-separated value (CSV) files.

To collect software measures, we used the Intel Energy Checker (IEC) SDK [25]. This SDK allows developers to insert counters in the application code, to record significant events and/or operational metrics (i.e. the number of queries executed by a DBMS, the time spent in a particular function, etc.). These counters can be exported through the same API, in order to be accessible from other applications at runtime. For power consumption data, we used the Intel Energy Server tool (ESRV). ESRV is part of the IEC SDK and works under the same principle. Basically, ESRV is a simple application able to interface itself with several power meters and DAQs and export the values read by those devices through a software counter, defined in the IEC API. The use of this tool allows to record both software events/measures and power consumption information (both per-resource, through DAQs, and system-level, through WattsUp PRO) using the same software construct. This reduces noise due to format conversions and synchronization issues.

To collect resource usage data, we used Dstat[5] for Linux/Unix. Dstat allows to combine the output of various resource monitoring tools commonly used in Unix environments (*vmstat, iostat, netstat, ifstat*). In particular, we gathered the following resources:

- CPU statistics (user time, system time, idle time and more)
- Disk statistics (read/write)
- I/O statistics (read/write)
- Memory statistics (paging, used memory, buffered/cached memory, available memory, swap)
- System load (1m, 5m, 15m)

[4]`https://www.wattsupmeters.com/secure/index.php`
[5]http://linux.die.net/man/1/dstat

- Network usage (packets sent/received)

The output was collected as CSV files with a granularity of 1 second.

3.4 Execution

3.4.1 Preparation

The context of our experiment is a **single-object study**: we apply a single object (i.e. a software practice) to a single subject (i.e. a software application). This choice has been made due to the intrinsic complexity of the practice application: as of now, energy-efficient software practices are described in literature as high-level guidelines, hence there are no formal specifications of how to apply a practice to an application. In this section, we describe how we implemented the practices and the assumptions we made.

For each practice, we developed three different scenarios:

- In the first scenario, that we called *vanilla*, the test application is benchmarked as-is, *without* introducing any code instrumentation. We developed this scenario to test our lab setting and to assess the impact of the instrumentation.
- The second scenario features the test application *with* code instrumentation, *before* applying the software practice under test.
- The third scenario features the test application *with* code instrumentation, *after* applying the software practice under test.

Practice 1: Use efficient queries

We implemented this practice using the MySQL Database Server software. As dataset, we used a full copy of the English Wikipedia articles as of 2008, which has a size of approximately 30GB. The English Wikipedia dataset has been obtained from the WikiMedia Foundation. We designed a simple query that iterates over all Wikipedia pages searching for text fragments. We disabled the MySQL internal cache, which could potentially be a confounding factor, by using the *SQL_NO_CACHE* keyword in the SQL statement. The application of the practice is simulated by issuing two different types of queries: one uses the *ORDER BY* keyword to order the results, the other one doesn't. We developed a benchmark that executes the SQL query 3 times. We decided not to control the duration of the benchmark for this practice: it varies according to the execution time of the queries. This allows us to assess the impact of the practice upon performance. Listings 3.1 and 3.2 show the SQL statements we used.

```
SELECT SQL_NO_CACHE a.old_id
FROM text a, revision b
WHERE a.old_id = b.rev_text_id
ORDER BY a.old_id;
```

Listing 3.1: Query before applying the practice

```
SELECT SQL_NO_CACHE a.old_id
FROM text a, revision b
WHERE a.old_id = b.rev_text_id
```

Listing 3.2: Query after applying the practice

Practice 2: Put application to sleep

We implemented this practice using a local installation of the Apache WebServer software v.2.2.25. Actually, the application already makes use of the *Put application to sleep* practice when waiting for an HTTP request. We modified the WebServer source code removing every call to the *sleep()* function in the body of the request handling procedure. This modified version represents the subject without the application of the practice. For benchmarking, we used the *ab* utility (Apache Benchmark) included in the WebServer package, with the following parameters:

ab -kc 50 -t 300 -n 5000000 http://localhost/

This configuration issues up to 5000000 requests, with a maximum of 50 concurrent requests and a time limit of 5 minutes (300 seconds). This allows us to control the length of the experiment and extract performance statistics: as opposed to the previous practice, where the number of query was fixed and the execution time was not, in this case the number of requests varies (the WebServer is not able to process 5 million requests in 5 minutes) but the execution time is fixed.

3.4.2 Data Collection and Analysis

For each scenario (*vanilla, before, after*) we performed 10 different executions. During each execution, we collected resource usage data through the CSV output of the *dstat* tool, energy usage through the ESRV tool (at resource level) and the WattsUp PRO meter (at system level) and software execution measures through the IEC API. We carefully checked the timestamps between our different logs to ensure synchronization. As shown in Figure 3.1, all the logs were collected

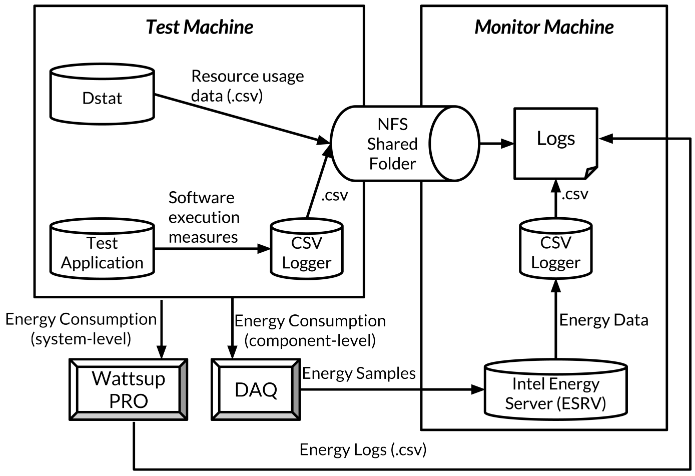


Figure 3.1: Experiment execution.

on a separate *monitor* machine, to minimize the measurement overhead on the test machine. For this reason, the energy meters were electrically connected to the test machine, but the data channels of the meters (i.e. USB cables of the DAQs and the WattsUp PRO) were connected to the monitor machine collecting the data samples. As regards the software measurements and the resource usage data, the *dstat* tool and the IEC API calls were performed on the test machine, but the output CSV logs were remotely written on an NFS shared folder located on the monitor machine.

The analysis of the data was performed using the R software for statistical computing[6]. We applied the following analysis techniques on the data (most of them are described in [86], references provided otherwise):

- Descriptive statistics (e.g. mean, median)
- Shapiro-Wilk test of normality [116]
- Correlation analysis using Spearman's rank coefficient
- Wilcoxon signed-rank test for assessing the impact of the practice
- Effect size computation using Cohen's *d*, Hedges' *g* and Vargha-Delaney A measure [130]

[6]`http://www.r-project.org/`

We choose a significance level $\alpha = 0.05$ for all of our tests, i.e. we accept a 5% chance of type I error (rejecting the null hypothesis when it is actually true). When evaluating correlations, due to the high number of comparisons we perform (21 comparisons for system-level analysis, 168 for resource-level analysis) we applied the Bonferroni correction to our significance level. Hence, at system level we have

$$\alpha_s = 0.05/21 \approx 0.002 \tag{3.1}$$

while at resource-level we have

$$\alpha_c = 0.05/168 \approx 0.0003 \tag{3.2}$$

All the raw data, the reports and the R scripts reproducing the reports are publicly available online[7].

3.5 Threats to Validity

Before reporting our experimental results (in Section 3.6), in this section we present the threats to validity and their mitigation. Our aim is of illustrating the premises and the assumptions behind our experimentation, hence increasing the readability of the experiment results. The classification of the threats follows the one by Cook and Campbell [24].

3.5.1 Conclusion Validity

Threats to conclusion validity affect the statistical significance of the findings. In our experimentation, we identify the following conclusion validity threats:

- *Reliability of measures.* When performing energy consumption analysis, the precision and accuracy of the measurement equipment is of utmost importance. For this reason, we performed our measurement in the SEFLab, a state-of-the-art laboratory purposely built to perform energy consumption analysis. We also collaborated with the staff and technicians who setup the lab in order to ensure the highest measurement quality (see Acknowledgements).

- *Reliability of treatment implementation.* As mentioned in Section 3.4, the application of the best practices to our application subjects is a complex process that cannot be standardized or automated, as of the current way practices are documented (this is also a threat to construct validity, see

[7]`http://www.s2group.cs.vu.nl/wp-content/uploads/2014/07/green-practices-online-package.zip`

below). Hence, we cannot guarantee that a different implementation would give similar results. To mitigate this threat, the implementation of the practice was performed by two different researchers and its meaningfulness was cross-checked with experts in the field.

3.5.2 Internal Validity

Threats to internal validity affect the causal interpretation of our results. We identify the following threats to internal validity:

- *Treatment assignment.* As we stated in Section 3.3, our empirical study is a *quasi-experiment*, for feasibility reasons: the assignment of a practice to a single software application is an operation that cannot be automated nor randomized at the present time. We are aware that this prevents us to fully establish causation. Part of our future efforts (see Section 3.8) will be devoted to generalize the practices in order to make them automatically applicable on multiple products, hence enabling randomized assignment.

- *Code instrumentation.* Due to the usage of the IEC API (see Section 3.3), we had to insert several additional calls in our application subjects. That might result in a confounding factor. To mitigate this threat, we also performed a benchmark of each application before performing instrumentation (*vanilla* scenario). This allowed us to estimate the impact of the instrumentation and take it into account.

3.5.3 Construct Validity

Threats to construct validity affect the relationship between theory and observation. Our main threat to construct validity regards the *operational explication of constructs*, meaning that the practices are not formalized in a standard and objective way, hence their translation into operational constructs is subject to interpretation. To mitigate this threat, we documented the practices to the best of our knowledge and we provided references of their sources. We seek, however, for researchers to challenge our interpretation and provide further examples to improve our knowledge base on best practices for software energy efficiency.

3.5.4 External Validity

Threats to external validity affect the generalization of our findings. For our experimentation, we identified the following external validity threats:

- *Subject selection.* For feasibility reasons, also due to the complexity of the application of a practice, we designed a single-object study, so we selected only two software applications for our study. Accordingly, we cannot guarantee that our subjects are representative of the whole population. To mitigate this risk, we chose our application subjects to be as representative as possible, being them widely-used open source software applications.

- *Experimental setting.* We conducted our experiments in a controlled environment. Hence, we cannot guarantee that our results would be the same in a different setting, e.g. the production environment of a company. However, the test machine and the application versions were as up-to-date as possible and they are widely used in industrial settings too.

3.6 Results

In this section, we present the results of our experiment. For each practice, we present the results of hypothesis testing and other relevant findings. We also discuss the results for each practice in detail.

3.6.1 Practice 1: Use Efficient Queries

Hypothesis Testing

In Table 3.5 we summarize the results for hypothesis testing. The Wilcoxon signed-rank test shows that the application of the practice induces a significant decrease in energy consumption (*Z=-3.915, p-value=0.00009*). Thus we can safely reject the null hypothesis.

Table 3.5: Practice 1: Results of hypothesis testing (*energy* consumption)

	Before	After
Median	47.85	35.82
Mean	43.83	35.82
% Diff.	-25.1 %	
Wilcoxon's Z	-3.915	
Cohen's d	-140.206 (large)	
Hedges' g	-134.282 (large)	
Vargha & Delaney's A	0 (large)	

Discussion on Practice 1

Before and after the application of the practice, Spearman's ρ correlation coefficient calculation returned different results in 15 over 21 pairs of resource–energy variables at system level (all p-values $< \alpha_s$, see Equation 3.1) and in 88 over 168 pairs of resource–energy variables at resource-level (all p-values $< \alpha_c$, see 3.2). In Table 3.6 we report all the significant correlation coefficients at system level. In Table 3.7, for the sake of brevity, we report only the resource-level coefficients that had a significant variation ($\Delta\rho > 0.4$) before and after applying the practice.

Table 3.6: Practice 1: Resource usage analysis at system-level

u	P	$cor(u_{r0}, P_0)$	$cor(u_{r1}, P_1)$
CPUusr	Watts	0.673	0.715
CPUsys	Watts	0.643	-0.163
CPUidl	Watts	-0.189	-0.319
CPUwai	Watts	-0.691	-0.632
CPUsiq	Watts	0.108	0.264
DSKread	Watts	-0.489	0.04
IOread	Watts	-0.696	-0.688
LOAD1m	Watts	-0.087	-0.177
LOAD5m	Watts	-0.079	-0.143
LOAD15m	Watts	-0.06	-0.092
MEMbuff	Watts	0.045	0.127

Table 3.7: Practice 1: Most significant results of resource usage analysis at component-level

u	P_r	$cor(u_{r0}, P_{r0})$	$cor(u_{r1}, P_{r1})$
CPUusr	MB.5V..Watts.	0.043	0.583
CPUsys	HDD1.5V..Watts.	-0.553	0.156
IOread	HDD1.5V..Watts.	0.618	0.177
IOread	MEM.12v..Watts.	-0.076	-0.628
IOread	MB.5V..Watts.	-0.07	-0.659

Our hypothesis testing confirms that our first practice, *Use efficient queries*, is successful in increasing energy efficiency. However, some additional considerations need to be done. As emerges from Table 3.8, the decrease in power consumption is significantly lower than energy, in percentage terms. This indicates that after the application of the practice, there is also an improvement in performance, which is reasonable due to the missing *ORDER BY* clause. Indeed, we report a significant difference in execution time: before the practice, we measured an average of 257 seconds per query, while after the practice the average time per query was 200 seconds.

The decrease in power consumption also indicates a different usage of the resources, as emerges from the results of RQ2: after applying the practice, we can observe a direct correlation rising between CPU activity and motherboard/disk consumption, that indicates a more energy-proportional behavior. This behavior becomes evident when analyzing the relationship between the CPU activity and the system-level power consumption, as shown in Figure 3.2.

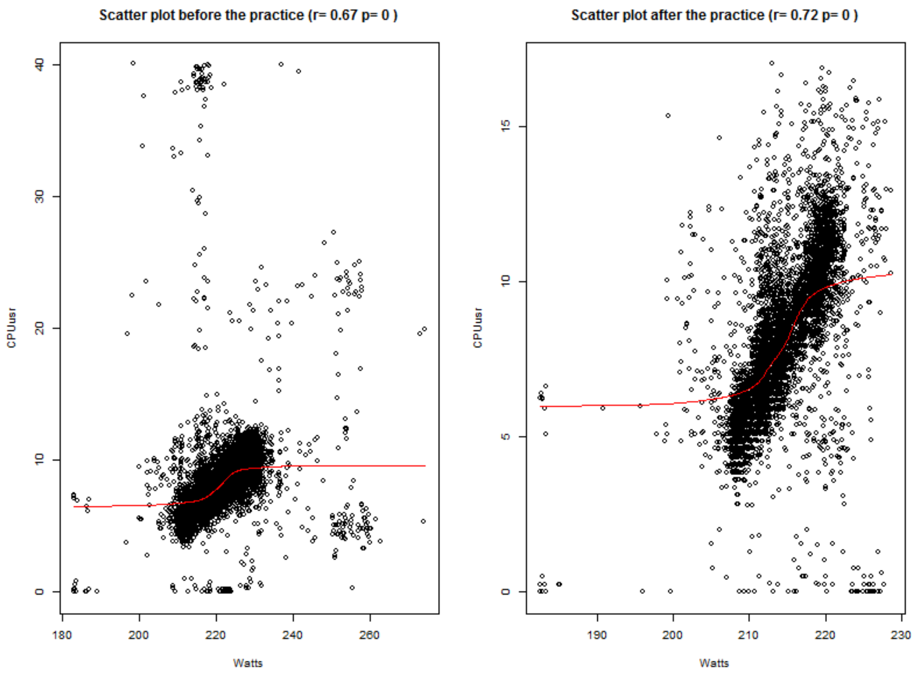


Figure 3.2: Scatter plots of the variables CPUusr and Watts before and after practice 1 was applied.

Another interesting insight regards the relationship between the I/O operation and power consumption: before the practice, there was no significant correlation, while after we observe negative correlation coefficients. This might be due to the fact that I/O activity is typically less power–intensive, as the most power consuming resource, the CPU, is inactive. Applying the practice reduces I/O activity by removing the *ORDER BY* clause which translates in less I/O read and writes, hence the negative impact of I/O over the overall power consumption is more evident, also due to the reduced execution time. This is also supported by the enhanced proportionality between energy and memory usage in Table 3.6.

Table 3.8: Practice 1: Effect size analysis (*power* consumption)

	Before	After
Median	221.22	214.06
Mean	219.64	213.47
% Diff.	-2.81 %	
Wilcoxon's Z	-3.468	
Cohen's d	-1.775 (large)	
Hedges' g	-1.700 (large)	
Vargha & Delaney's A	0.09 (large)	

3.6.2 Practice 2: Put Application to Sleep

Hypothesis Testing

In Table 3.9 we summarize the results for hypothesis testing. The Wilcoxon signed-rank test shows that the application of the practice induces a significant decrease in energy consumption (*Z=-3.929, p-value=0.00008*). Thus we can safely reject the null hypothesis.

Table 3.9: Practice 2: Results of hypothesis testing (*energy* consumption)

	Before	After
Median	24.11	22.06
Mean	24.10	22.06
% Diff.	-8.48%	
Wilcoxon's Z	-3.929	
Cohen's d	-42.069 (large)	
Hedges' g	-40.292 (large)	
Vargha & Delaney's A	0 (large)	

Discussion on Practice 2

Before and after the application of the practice, Spearman's ρ correlation coefficient calculation returned different results in 7 over 21 pairs of resource–energy variables at system level (all p-values $< \alpha_s$, see 3.1) and in 55 over 168 pairs of resource–energy variables at resource level (all p-values $< \alpha_c$, see 3.2). In Table 3.10 we report all the significant correlation coefficients at system level. In Table 3.11, for the sake of brevity, we report only the resource-level coefficients that had a significant variation ($\Delta\rho > 0.4$) before and after applying the practice.

The results of hypothesis testing, as for Practice 1, confirm the usefulness of Practice 2 in improving energy efficiency. That being said, the two practices affect energy consumption and resource usage in different ways.

First of all, for Practice 2 there is almost no difference in the impact between energy and power consumption, as shown in Table 3.12. This indicates a less evident impact on performance: indeed, benchmarks before the practice indicated an average time per request of 0.210 milliseconds, as opposed to 0.196 milliseconds after the practice, hence a mere 6% improvement. However, the average energy consumed per request is 16.89 pWh before the practice and 14.41 pWh after, with a reduction of 14%. Thus, energy savings are not only due to a shorter execution time.

Table 3.10: Practice 2: Results of resource usage analysis at system-level

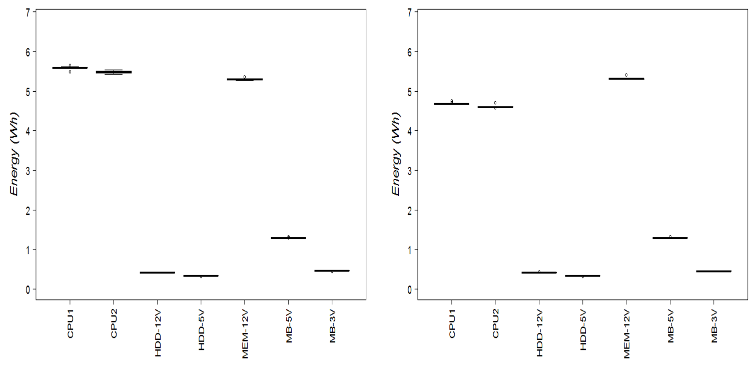
u	P	$cor(u_{r0}, P_0)$	$cor(u_{r1}, P_1)$
CPUusr	Watts	0.691	0.886
CPUsys	Watts	0.159	0.761
CPUidl	Watts	-0.736	-0.905
CPUsiq	Watts	0.527	0.423
MEMused	Watts	-0.727	-0.849
MEMcach	Watts	-0.43	-0.392
MEMfree	Watts	0.596	0.593

Table 3.11: Practice 2: Most significant results of resource usage analysis at component-level

u	P_r	$cor(u_{r0}, P_{r0})$	$cor(u_{r1}, P_{r1})$
CPUusr	CPU1.12V..Watts.	0.144	0.773
CPUusr	CPU2.12V..Watts.	0.078	0.8
CPUusr	MB.5V..Watts.	0.086	0.631
CPUsys	MEM.12v..Watts.	0.117	0.655
CPUidl	CPU1.12V..Watts.	-0.158	-0.793
CPUidl	CPU2.12V..Watts.	-0.075	-0.815
CPUidl	MB.5V..Watts.	-0.095	-0.655
MEMused	CPU1.12V..Watts.	-0.111	-0.771
MEMused	CPU2.12V..Watts.	-0.103	-0.788
MEMfree	CPU1.12V..Watts.	0.072	0.553
MEMfree	CPU2.12V..Watts.	0.111	0.55

The correlation analysis shows us clearly that, as for Practice 1, applying this practice leads to a more energy–proportional behavior, as all CPU–power coefficients increase (*CPUidl* represents the time spent by the CPU in idle time, which is negatively correlated with power, as expected). This phenomenon becomes evident by looking at the scatter plot in Figure 3.4.

However, as shown in the box-plot of Figure 3.3, memory (MEM-12V) also plays an important role: the average energy consumed by the memory amounts to 5.297 Wh per run, as opposed to the 5.47 and 5.58 Wh consumed by the two CPUs (CPU1 and CPU2, respectively). This shows that when optimizing software for energy efficiency, we need to take into account memory usage as



(a) (b)

Figure 3.3: Energy consumed per resource before (a) and after (b) applying Practice 2.

Table 3.12: Practice 2: Effect size analysis (*power* consumption)

	Before	After
Median	297.06	272.63
Mean	296.62	272.06
% Diff.	-8.28%	
Wilcoxon's Z	-4.548	
Cohen's d	-16.890 (large)	
Hedges' g	-16.176 (large)	
Vargha & Delaney's A	0 (large)	

well.

3.7 Reflection

In this section, we provide the reader with some reflection points that emerge from our results.

The first point regards the main outcome of this study: the impact of the best practices for software energy efficiency. We reported very significant energy consumption reductions, up to 25%, that show the relevance of this research and more in general of software energy efficiency. This relevance even increases when

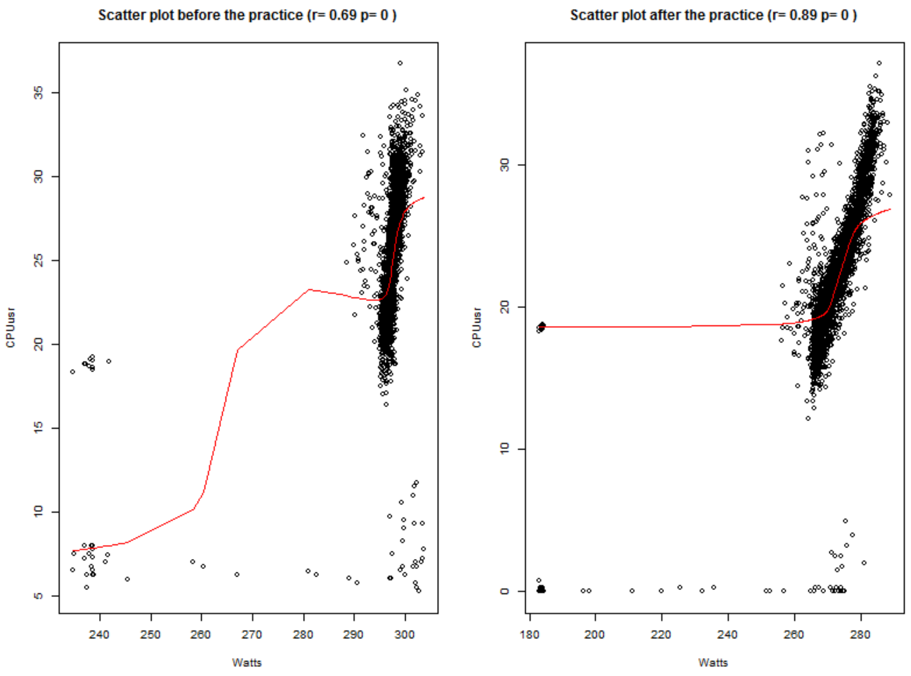


Figure 3.4: Scatter plots of the variables CPUusr and Watts before and after practice 2 was applied.

considering emerging contexts such as Cloud Computing and Big Data, where software applications run in countless instances, hence their energy efficiency becomes crucial. Another relevant context is High Performance Computing: as we approach the so-called Exascale Computing era [123], when the power consumption of HPC systems is predicted to be in the order of tens of MWs, energy efficiency will be among the top priorities of the sector.

Indeed, the related work we summarized in Section 3.2 reports a number of findings in heterogeneous contexts, sometimes presented as suggestions/guidelines for developers. In this work, we present two best practices, documented with a structured template [46], and we assess their impact. The template gives an added value in terms of reusability of the practices, while the characterization of their impact helps developers in planning for reaching a specific level of energy efficiency, provided the compatibility of the practice with the application domain and other possible constraints.

If we look at the impact of software energy efficiency on the software engineering process, it introduces a new concern that potentially affects all phases of software development. Reusable best practices guide software architects and developers in making *energy efficient* design decisions and implementation choices, hence ensuring software to be energy efficient in the first place.

Finally, our experimental design and setting represents an important part of our scientific contribution. A dedicated laboratory environment for assessing software energy efficiency represents the starting point of a sound methodology for research in software energy efficiency, as well as a testbench, when software testing for energy efficiency will be common practice, as it is now for other software qualities. The tools and analysis methods we adopted fit into a more general, reusable framework for energy efficiency in software engineering, that we will further develop in the remainder of this thesis.

3.8 Conclusions

In this contribution, we presented the empirical validation of two best practices for energy-efficient software development, selected from a collection of practices elicited from both academic and industrial sources.

For our study we selected two best practices, *Put application to sleep* and *Use efficient queries*, and we applied them on well-known and widely adopted open source products. This chapter answers RQ 2, namely: "*What is the impact of using best practices for software energy efficiency?*" The results of our empirical experimentation show a significant and consistent impact of adopting both practices in increasing the energy efficiency of the selected subjects, namely the Apache WebServer and the MySQL Database Server. A more detailed analysis also showed that the practices significantly alter the resource usage pattern of the applications, inducing a more energy-proportional behavior after their application.

We have set up a wiki[8] to distribute our practices, along with a structured template for their documentation where we will include the results of our empirical validation, so that practitioners can learn how adopting a practice can improve the energy efficiency of their product. In the next future, we plan to set up a regular experimentation activity on all the practices collected so far, formalizing our current experimental design and incrementally building a general framework for empirical research on energy-efficient software. Other future efforts will be aimed at generalizing the energy-efficient software practices, by abstracting the general concepts behind them and eventually providing automated refactoring procedures to apply them on existing products. This will speed up our experimentation and hence increase the amount (and maturity) of evidence

[8]https://wiki.cs.vu.nl/green_software/index.php/Main_Page

we can gather through our methodology. For this purpose, we developed a new approach for empirical experimentation in energy efficiency, that is presented in Chapter 7. In the next part of the thesis, we will instead analyze how energy efficiency can be addressed in the early phases of the software life cycle, i.e. the architectural phase.

4 Energy Efficiency in Cloud Software Architectures - A Systematic Literature Review

In the previous chapters, we focused on how to support developers to increase the energy efficiency of existing applications. This chapter further elevates the level of abstraction, by analyzing the architectural implications of addressing energy efficiency. For this purpose, we look at Cloud-based software architectures. Although cloud computing is often considered as an energy–efficient technology, the implications of cloud–based software on energy efficiency lack scientific evidence. At the same time, energy efficiency is becoming a crucial requirement for cloud service provisioning, as energy costs significantly contribute to the Total Cost of Ownership of a data center. In this chapter, we present the design and results of a Systematic Literature Review on energy efficiency in Cloud Software Architectures. This study provides us the answer to RQ 3a.

4.1 Introduction

Cloud computing infrastructures are often described as an energy-efficient technology [13]. In principle, improving data center utilization by virtualizing resources is a way to save energy. However, even if it has been proven that cloud technologies provide benefits in terms of energy savings, this factor is not adequately exploited as an added value for cloud service provisioning.

Nowadays, energy efficiency is starting to be considered as a Service-Level Objective (SLO), i.e. a specific, measurable characteristic of a service, to be described as achievement values in Service Level Agreements (SLAs)[1]. An ex-

[1] `http://www.greenbiz.com/news/2009/01/12/energy-efficiency-new-sla`, last visited on June 12th, 2013

ample would be: "The energy bill of the client should be reduced by 20% in one year". Cloud service providers (CSPs) could benefit from representing energy efficiency as a SLO. However, in order to offer cloud services, providers rely on very complex software architectures. It is yet unclear, and possibly unexplored, what architecture characteristics do to positively or negatively influence energy efficiency, and if there are explicit or implicit reference architectures that can help in increasing energy efficiency.

Nowadays, the role of software in energy consumption is widely discussed among the scientific community, and a number of metrics for software energy efficiency has been proposed [15]. Our work tries to advance to the next step: whether it is possible to quantify the effects on energy consumption when adopting a certain software architecture, and what architectural solutions can be adopted to increase energy efficiency in cloud-based software.We performed a Systematic Literature Review (SLR) [63] to investigate the relationship between cloud-based software architectures and energy efficiency.

This chapter is structured as follows: Section 4.2 describes our review protocol in detail. Section 4.3 presents the results of a demographic analysis conducted on our primary studies. Section 4.4 provides insights about the state–of–the–art of energy efficiency in cloud software architectures. Section 4.5 gives an overview of the stakeholders for energy efficiency we identified during our research. In Section 4.6 we discuss the threats to validity that might affect our study. Section 4.7 concludes the chapter.

4.2 Review Protocol

This study is aimed at answering RQ 3a, namely:

RQ3a. *Are there software architectural solutions that address energy efficiency aspects?*

As we stated in Section 4.1, we focus specifically on cloud service provisioning, and how software architectural solutions can be adopted to achieve Service Level Objectives on energy efficiency. In order to answer our RQ, we followed a systematic literature review process. We performed a preliminary analysis of the research space, and we identified 306 hits (i.e. potentially related studies). We formulated a review protocol for our study, by defining a search query for academic databases and inclusion and exclusion criteria. Applying the protocol, we selected the primary studies for our research. We subsequently classified and analyzed these studies in order to extract relevant results.

In this section, we extensively describe our protocol, for the sake of reproducibility. All the main components of the protocol will be discussed: search strategy, study selection, data extraction, data analysis and traceability.

4.2.1 Search Strategy

We adopted *Google Scholar*[2] as our data source. We defined a query string by selecting the most appropriate keywords to answer our RQ. We selected five keywords: "software architecture", "cloud", "service", "SLO", "energy". Our query was defined after different steps, using the results of our preliminary analysis as pilot to test the coverage of the results. Namely, if one of the studies in our pilot was not retrieved by the query string, we refined it to add more keywords (typically, acronyms or alternative spellings, e.g. *"service level agreement"* vs. *"SLA"*.

The final query string was defined as follows:

"software architecture" AND cloud AND service AND "(energy OR power) efficiency" AND (SLA OR SLO OR "service level")

The query string was applied to titles, abstract and body of the studies, to enlarge the scope as much as possible. Our time range for the search went from 2000 to 2013. This period was chosen considering the relatively recent development of cloud computing technologies.

4.2.2 Study Selection

In order to select our primary studies, we defined a number of criteria for inclusion and exclusion. The criteria select papers in terms of their relevance to our RQ, but also in terms of scientific validity and language. In general, a study is selected if it fulfills all of the inclusion criteria, and excluded if it fulfills any of the exclusion criteria. Table 4.1 summarizes the Inclusion–Exclusion Criteria for our review protocol.

4.2.3 Data Extraction

We used an extraction form in order to retrieve and store relevant information about each primary study. Besides general information, the form records how energy efficiency is addressed and which architectural elements were identified in the presented solution. The extraction form is structured as follows:

- **Study Identifier**: provides an identifier for the study;
- **Study Title**: the publication title;
- **Study Type**: the publication type (i.e. journal article, conference article, thesis);

[2] `http://scholar.google.com/`

Criterion	*Rationale*
I1 *A study that directly proposes software architectures, architectural styles or strategies, or indirectly proposes them from a service provisioning perspective.*	We want to identify how software architectures affect energy efficiency, thus we need articles proposing software architectures, or indirectly proposing them from a service provisioning perspective.
I2 *A study that addresses energy efficiency as a quality attribute.*	We want to investigate whether energy efficiency is considered, by providers or experts, as a quality attribute for cloud services.
I3 *A study that is developed by either of academics and practitioners.*	Both academic and industrial solutions are relevant to this study.
I4 *A study that is published in software engineering/cloud computing field.*	Software engineering is our reference field, but cloud computing research can provide us an insight on what trends are set in terms of software architectures for cloud.
I5 *A study that is peer-reviewed.*	A peer-reviewed paper guarantees a certain level of quality and contains reasonable amount of content.
I6 *A study that is written in English.*	For feasibility reasons papers written in other languages than English are excluded.
E1 *A study that does not propose software solutions for energy efficiency.*	Traditionally, energy efficiency has been regarded as an hardware issue. We want to drive past this assumption and address the software impact of power consumption.
E2 *A study that does not imply any type of service provisioning.*	We are not interested in solutions that generally increase the energy efficiency of a datacenter, without having in mind how to provide an energy-efficient service to a customer.
E3 *A study that does not consider energy efficiency as a primary quality attribute.*	We are not interested in studies that consider energy efficiency a secondary concern.
E4 *A study that does not aim at optimizing the energy efficiency of the cloud computing infrastructure.*	Mobile devices often leverage cloud services by offloading computation tasks, in order to increase their battery life. Although this is an energy efficiency improvement, it is not relevant for the energy efficiency of the cloud computing infrastructure, thus we want to exclude these solutions from our study.

Table 4.1: Inclusion and Exclusion Criteria.

- **How energy efficiency is addressed**: a brief summary of how the presented solution addresses the energy efficiency of the cloud infrastructure;
- **Main architectural elements**: the main software elements of the solution.
- **Stakeholders**: stakeholders mentioned in the study that can be affected or involved in the architectural solution presented.
- **Validation**: whether the proposed solution has been validated in an Academic or Industrial setting, or no validation was performed. The validation is considered Academic when the article has been validated through a simulation or a test-bed. The validation is considered Industrial when the article reports a real case study (i.e. the proposed solution is already implemented in a software product).

4.2.4 Data Analysis

Our RQ investigates how cloud software architectures deal with energy efficiency issues. The aim of an SLR is to "identify, analyze and interpret all available evidence related to a specific research question" [64]. Hence, we do not aim at directly providing new reusable solutions or patterns, but rather we aim at classifying the existing body of knowledge in a systematic way.

To elicit this information, we adopted *coding*. Coding is a qualitative research method, commonly used in social sciences, that interprets data and organizes it in categories or families, using *codes*, i.e. words or short phrases. Coding allows to capture the fundamental information of qualitative data in a systematic way, and enables us to link it and discover patterns and trends [108].

Our first step was an exploratory study of the selected contributions, in order to define an initial set of codes (or "start-list" [84]). The start-list was built by analyzing reference literature in software architecture [12][55][96][27]. We arranged our codes in a conceptual three-level structure, shown in Figure 4.1 and defined as follows:

- *Strategy*: the high-level approach through which a software solution addresses energy efficiency;
- *Technique*: the instantiation, or enactment, of a strategy through a specific technical approach;
- *Component*: an individual architectural component that plays a defined role in the application of a technique.

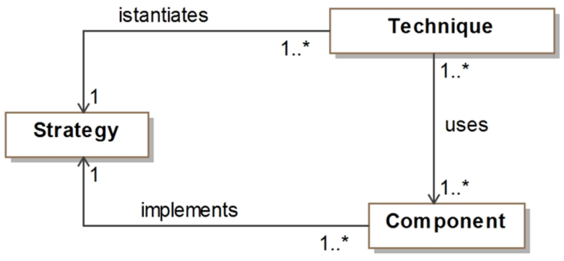


Figure 4.1: Conceptual structure of our codes.

The concept of architectural strategy [60], technique (or tactic) [12] and component [40] are very well known foundational concepts of software architecture and they are familiar to practitioners and experts in the field. By adopting this conceptual structure, we aim at communicating our findings more effectively to software architects.

Finally, our primary studies were iteratively analyzed by two researchers independently, refining the list at every iteration until general and unambiguous codes were identified.

4.2.5 Traceability

We recorded the reference information of the studies using JabRef[3], a software tool for reference management. JabRef manages references in BibTeX and many other formats, and also allows to link and embed full-texts. For every step of the review process, a different JabRef database file was created that contained the references of the studies analyzed in that step. Moreover, we used an Excel sheet to report the matching of the inclusion/exclusion criteria and the stage of the decision (title, abstract or full-text checking) for each study. As regards the traceability of our analysis, whenever a code was identified in a primary study we annotated the corresponding section of the full-text of the contribution. In this way, the systematic mapping we performed can be verified by independent reviewers. All the material is available on request.

4.3 Demographic Analysis

In Table 4.4 we present the results of the application of our review protocol. The search query of Section 4.2 identified 149 initial results in the Google Scholar

[3]`http://jabref.sourceforge.net/`, last visited on January 29th, 2014.

database. Then we went through the primary study selection process, divided into three phases: first, we checked the title against our inclusion/exclusion criteria, then we checked the abstract and finally the full-text. At each step, we were able to exclude a number of studies from our initial set and we finally ended up with 26 primary studies. Tables 4.2, 4.3 present the list of the primary studies we identified during our SLR.

Figure 4.2 shows the distribution of our primary studies over time. It can be noted that the topic of energy efficiency in cloud service provisioning has become a concern in the past couple of years, as our primary studies have been mostly published starting from 2011. This distribution is coherent with the cloud computing hype cycle documented by Gartner [120]: in 2011 cloud computing was on top of the "Peak of Inflated Expectations". This reflects the growth of publications on cloud computing in 2011 that we can observe in Figure 4.2. No primary studies were identified in 2013: however, this is most likely due to the fact that when the search was performed (June 2013) many contributions published in the first months of the year were probably not indexed yet.

The distribution of the type of the articles is as follows:

- 11 journal articles;
- 12 conference articles;
- 3 PhD theses.

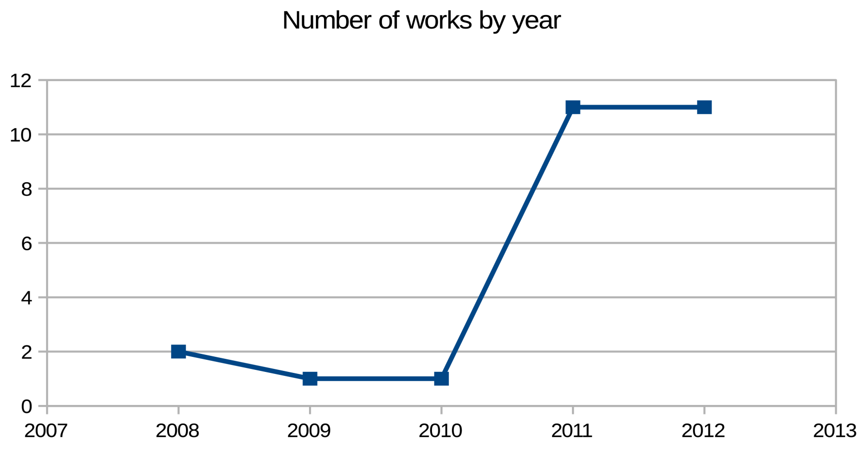


Figure 4.2: Number of primary studies selected per year.

Title	Authors	Year	Publication Type	Venue	Validation	Ref.
Application patterns for green IT	Rogers, D. and Homann, U.	2008	Journal	The Architecture Journal (MSDN)		[106]
Environmentally Sustainable Infrastructure Design	Curtis, L.	2008	Journal	The Architecture Journal (MSDN)		[26]
Taming energy costs of large enterprise systems through adaptive provisioning	Hedwig, M.	2009	Ph.D. Thesis		Academic	[50]
Self-optimization of the energy footprint in Service-Oriented Architectures	De Oliveira, J. et al.	2010	Conference	1st International Workshop on Green Computing Middleware (GCM'2010)	Academic	[28]
A Middleware framework for self-adaptive large scale distributed services	Chacin Martinez, P. J. et al.	2011	Ph.D. Thesis		Academic	[22]
Cloud management: Challenges and opportunities	Forell, T. et al.	2011	Conference	2011 IEEE International Parallel & Distributed Processing Symposium	Industrial	[36]
Configuration and Deployment Derivation Strategies for Distributed Real-time and Embedded Systems	Dougherty, B.P.	2011	Ph.D. Thesis		Academic	[30]
Decomposing Workload Bursts for Efficient Storage Resource Management	Lu, L. et al.	2011	Journal	IEEE Transactions on Parallel and Distributed Systems	Academic	[80]
Energy aware cloud application management in private cloud data center	Xu, L. et al.	2011	Conference	International Conference on Cloud and Service Computing (CSC)	Industrial	[141]
Energy Efficiency in integrated IT and optical network infrastructures: The GEYSERS approach	Tzanakaki et al.	2011	Conference	IEEE Conference on Computer Communications Workshops		[129]
Green Cloud Framework for Improving Carbon Efficiency of Clouds	Garg, S. K. et al.	2011	Conference	17th International Conference on Parallel Computing (EURO-PAR 2011)		[39]
Model-based self-adaptive resource allocation in virtualized environments	Huber, N. et al.	2011	Conference	6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2011)	Academic	[53]
Runtime Variability Management for Energy-efficient Software by Contract Negotiation	Götz, S. et al.	2011	Conference	Proceedings of the 6th International Workshop Models@run.time (MRT 2011)		[43]
Self-Aware Software and Systems Engineering: A Vision and Research Roadmap	Kounev, S.	2011	Journal	GI Softwaretechnik-Trends	Academic	[67]
Supporting energy-driven adaptations in distributed environments	Noureddine, A. et al.	2011	Conference	Proceedings of the 1st Workshop on Middleware and Architectures for Autonomic and Sustainable Computing	Academic	[90]

Table 4.2: Overview of the primary studies.

Title	Authors	Year	Publication Type	Venue	Validation	Ref.
A BDI agent-based approach for Cloud Application autonomic management	Xu, L. et al.	2012	Conference	IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom 2012)		[142]
A review of middleware approaches for energy management in distributed environments	Noureddine, A. et al.	2012	Journal	Software: Practice and Experience		[91]
A Survey on Energy Efficient Server Consolidation Through VM Live Migration	Sekhar, J. et al.	2012	Journal	International Journal of Advances in Engineering & Technology		[112]
A service framework for energy-aware monitoring and VM management in Clouds	Katsaros, G. et al.	2012	Journal	Future Generation Computer Systems	Academic	[59]
An energy aware framework for virtual machine placement in cloud federated data centres	Dupont, C. et al.	2012	Conference	2012 Third International Conference on Future Energy Systems (e-Energy 2012)	Academic	[31]
Using Queuing Theory for Controlling the Number of Computing Servers	Sevalnev, M. et al.	2012	Conference	Proceedings of the 3rd International Conference on Green IT Solutions (ICGREEN 2012)	Academic	[115]
Cloud Engineering is Search Based Optimization too	Harman, M. et al.	2012	Journal	Journal of Systems and Software		[49]
Cloud federation in a layered service model	Villegas, D. et al.	2012	Journal	Journal of Computer and System Sciences		[132]
CompatibleOne: Designing an Energy Efficient Open Source Cloud Broker	Carpentier, J. et al.	2012	Conference	Second International Conference on Cloud and Green Computing (CGC 2012)	Academic	[21]
Green cloud computing schemes based on networks: a survey	Xiong, N. et al.	2012	Journal	IET Communications	Academic	[140]
Towards a Cloud Infrastructure for Energy Informatics	Wu, Z. et al.	2012	Journal	Sprouts: Working Papers on Information Systems	Academic	[139]

Table 4.3: Overview of the primary studies.

Step	***Removed***	***Remaining***
Initial search results	N/A	149
Title checking	10	139
Abstract checking	78	61
Full-text checking	35	**26**

Table 4.4: Overview of the selection process.

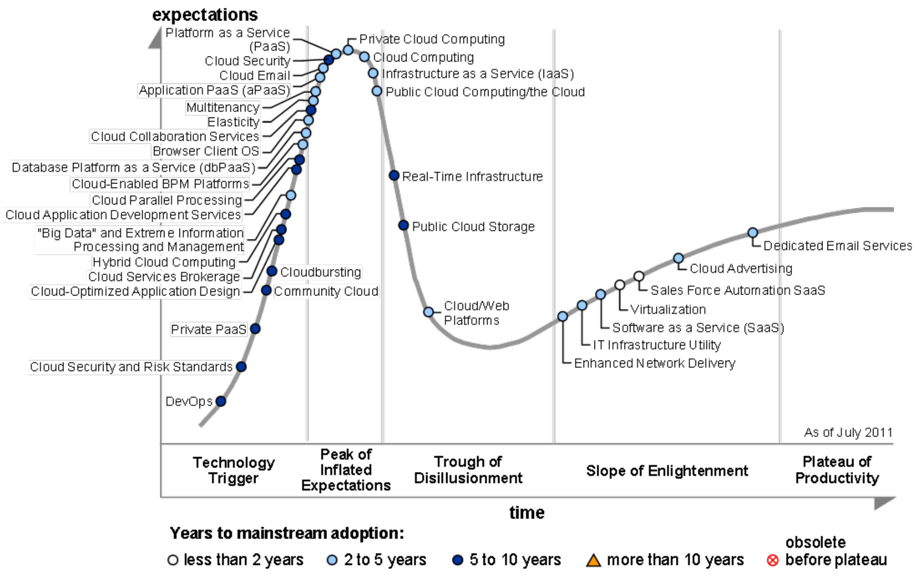


Figure 4.3: Cloud Computing hype cycle, 2011 [120].

As regards the validation of the solutions presented in the primary studies, we found that:

- 14 studies present an *Academic* validation;
- 2 studies present an *Industrial* validation;
- 10 studies do not validate the presented solution.

This testifies the low level of maturity of this topic. The lack of industrial validation indicates that the state-of-the-practice of energy efficiency in cloud software architectures has still to be determined.

4.4 Energy Efficiency in Software Architectures

Our results provide many insights on how energy efficiency is addressed by software architectures. As introduced in Section 4.2.4, results have been classified in terms of Strategies, Techniques and Components.

4.4.1 Strategies

We identified three strategies in the primary studies, namely:

- **Energy Monitoring**: this strategy is identified when some components of the software architecture of the presented solution are devoted at monitoring energy consumption;
- **Self Adaptation**: this strategy is identified when some components of the software architecture of the presented solution enable the possibility of adapting the software behaviour in order to increase energy efficiency;
- **Cloud Federation**: this strategy is identified when the software architecture of the presented solution comprehends the possibility to "lease" or "negotiate" the usage of cloud services from other providers according to energy consumption requirements.

The following list enumerates the occurrences of the different strategies and their combinations among the articles. A graphical overview is shown in Figure 4.4.

- *Energy Monitoring (alone)*: identified in 1 primary study.
- *Self-Adaptation (alone)*: identified in 11 primary studies.
- *Cloud Federation (alone)*: identified in 3 primary studies.
- *Energy Monitoring + Self-Adaptation*: identified in 8 primary studies.
- *Energy Monitoring + Cloud Federation*: absent.
- *Self-Adaptation + Cloud Federation*: identified in 1 primary study.
- *Energy Monitoring + Self-Adaptation + Cloud Federation*: identified in 2 primary studies.

Among the three architectural strategies we identified, Self-Adaptation is the most adopted (i.e. identified 21 times). Energy Monitoring is almost never

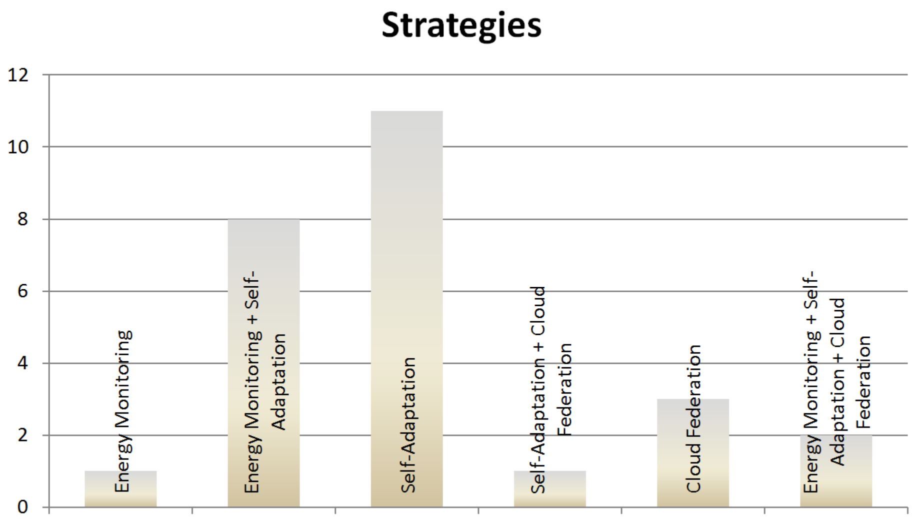


Figure 4.4: Distribution of strategy combinations among the primary studies.

adopted in isolation, but most of the time (i.e. 10 out of 11 studies) it is combined with Self-Adaptation, as emerges from Figure 4.4. This suggests that Energy Monitoring techniques are usually adopted as enablers for Self-Adaptation techniques, providing necessary information to drive the adaptation process. The low adoption of Cloud Federation techniques might be due to the fact that multi-cloud environments are still uncommon, mostly due to standardization and security concerns [35, 16].

4.4.2 Techniques

For each strategy, we identified a number of techniques, through which the strategy is enacted. In Figure 4.5 we show the distribution of the techniques among the primary studies. A more detailed description can be found in Table 4.5. As for strategies, techniques are not applied in isolation: in all primary studies, more than one technique per study is applied.

From Figure 4.5 a clear trend emerges: Consolidation and Workload Scheduling are by far the most adopted techniques (i.e. identified 11 and 18 times of 26 studies, respectively). Both of these techniques are very popular in cloud systems also for performance purposes, so this result is realistic and not surprising. Another interesting finding is that many techniques exhibit dependencies between each other. For example, we observed that scheduling algorithms or VM

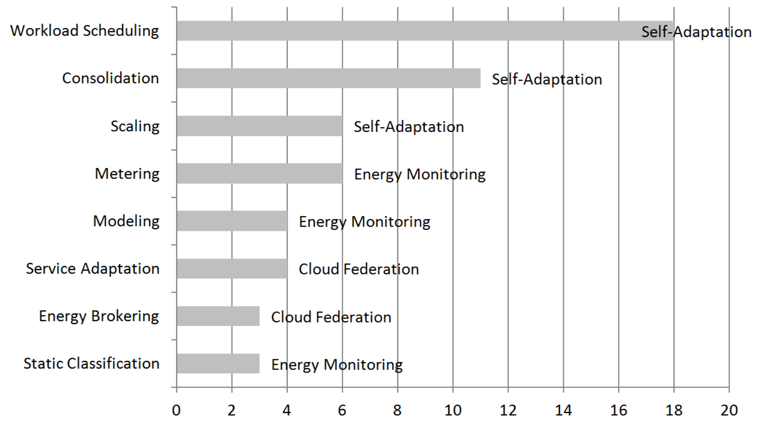


Figure 4.5: Distribution of techniques among the primary studies.

allocation processes are typically driven by components responsible for monitoring the infrastructure/system energy consumption. This implies that Workload Scheduling and Consolidation techniques depend on Metering and/or other Energy Monitoring techniques. Another dependency lies between the two Cloud Federation techniques: Service Adaptation needs an Energy Broker in order to retrieve the energy information of services and perform the service switching.

Strategy	Techniques	Description	Occ.	References
Energy Monitoring	Metering	Power consumption real-time monitoring through external power meters.	6	[21] [26] [36] [59] [90] [141]
	Static Classification	Energy classification of software based upon the power consumption specifications of the hardware components.	3	[129] [26] [43]
	Modeling	Power consumption on-line estimation using predictive models.	4	[28] [30] [31] [90]
Self-Adaptation	Scaling	Software is able to scale down in case of low requests or usage, to save energy.	6	[22] [30] [106] [49] [59] [141]
	Consolidation	In virtualization scenarios, the possibility to regroup VMs sparse among many servers, to reduce the number of active machines.	11	[21] [106] [31] [49] [59] [67] [91] [112] [140] [141] [142]
	Workload Scheduling	Some components of the software architecture are devoted to manage and schedule the workload of the computational units.	18	[129] [22] [28] [30] [36] [43] [49] [50] [53] [59] [67] [80] [90] [112] [115] [140] [141] [142]
Cloud Federation	Energy Brokering	The software architecture exposes their services together with their energy consumption information.	3	[36] [39] [132]
	Service-Adaptation	Switching functional services depending on their energy consumption.	4	[36] [132] [139] [140]

Table 4.5: Overview of the identified architectural techniques for energy efficiency.

4.4.3 Components

For each strategy, we identified the software architecture components primarily responsible for its implementation. In Figure 4.6 we present their distribution among the primary studies. A more detailed description can be found in Table 4.6. The relationship between components and techniques is many–to–many: a technique uses a number of components and each component can be used in more than one technique.

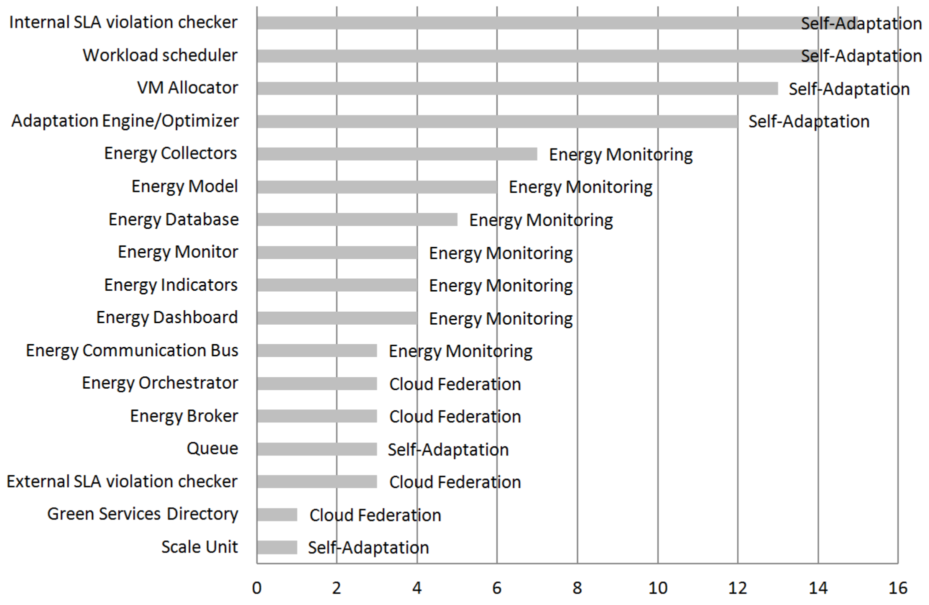


Figure 4.6: Distribution of components among the primary studies.

The high frequency of Workload Scheduling and Consolidation techniques is also reflected in terms of components, as shown in Figure 4.6: as expected, the Workload Scheduler and the VM Allocator are the second and third most frequent component identified (i.e. 14 and 13 times out of 26 studies, respectively). There are cases in which the component is found outside of its most typical technique: that is because in those cases, the component plays a role that does not implement that technique. For example, in [30] the VM Allocator is not used in a Consolidation technique but rather in a particular case of a Scaling technique.

The high number of occurrences of the SLA Violation Checker is one of our key findings. In particular, they are present in both of the primary studies that received an industrial validation. This suggests that the trade-off between energy efficiency and other software quality aspects appears to be a major architectural concern. In particular, it is relevant to notice the difference between the Internal and External component: the Internal SLA Violation Checker monitors the fulfillment of the SLAs when performing Self-Adaptation techniques (i.e. Scaling, Consolidation or Workload Scheduling) so it typically has to pose constraints to the internal adaptation process. The External SLA Violation Checker instead enforces that when negotiating services between different providers, the resulting service composition matches the required quality of service for a certain task. That is, the External SLA Violation Checker poses constraints to the service composition process.

Strategy	Components	Role	Occ.	References
Energy Monitoring	Energy Dashboard	Provides users or managers with software energy consumption information.	4	[21] [26] [36] [59]
	Energy Database	Stores energy consumption information.	5	[129] [21] [26] [59] [90]
	Energy Indicators	"Rate" or classify software behaviour, or provide real-time metrics upon energy consumption.	4	[28] [31] [36] [59]
	Energy Collectors	Retrieve and collect energy information from hardware or software sensors.	7	[21] [26] [28] [36] [59] [90] [141]
	Energy Communication Bus	Provide a common interface for collectors to the energy database.	3	[21] [26] [59]
	Energy Model	Estimate or predict the power consumption of a software application in real-time.	6	[28] [30] [31] [43] [90] [141]
	Energy Monitor	Monitor the energy consumption of (a part of) the software system.	4	[28] [31] [90] [141]
Self-Adaptation	Adaptation Engine/Optimizer	Find an optimal solution to an objective function modeling the energy efficiency of the system.	12	[129] [22] [28] [43] [49] [50] [53] [59] [67] [80] [141] [142]
	Workload Scheduler	Define, schedule and assign workloads to computational units.	14	[129] [22] [36] [43] [49] [50] [53] [59] [67] [80] [112] [140] [141] [142]
	Scale Unit	A defined set of IT resources that represents a certain scaling level.	1	[106]
	Queue	Organize items (services, VMs, jobs) in different orders of priority according to energy consumption.	3	[30] [80] [115]
	VM Allocator	In virtualized environments, migrate and displace VMs on servers.	13	[129] [21] [106] [30] [31] [49] [59] [67] [91] [112] [140] [141] [142]
	Internal SLA violation checker	Check and ensure the fulfillment of SLAs (NOTE: in this case the checker evaluates the violation of internal services towards external clients).	15	[22] [106] [28] [30] [31] [43] [49] [50] [53] [59] [67] [91] [80] [141] [142]
Cloud Federation	Energy broker	Provides access to energy efficient services.	3	[36] [39] [132]
	Energy Orchestrator	In SOA contexts, switch services in case of relevant differences in their energy efficiency.	3	[132] [139] [140]
	Green Service Directory	Provides a listing of all available services with energy consumption information.	1	[39]
	External SLA violation checker	Check and ensure the fulfillment of SLAs (NOTE: in this case the checker evaluates the violation of external services towards internal clients).	3	[31] [36] [132]

Table 4.6: Overview of software components for energy efficiency.

4.5 Stakeholder Overview

An important part of our data analysis focuses on the stakeholders that could have been affected or interested by the architectural solution introduced in the primary studies. Our aim is to identify stakeholders for energy efficiency, whose concerns can be targeted as an architectural concern. In Table 4.7 we show the stakeholders we identified, along with their definition and the criteria behind their identification. They are mentioned with the following frequency:

- *End-User*: mentioned in 6 primary studies.
- *Service Provider*: mentioned in 10 primary studies.
- *System Architect*: mentioned in 13 primary studies.
- *Infrastructure Manager*: mentioned in 12 primary studies.

Stakeholder	**Definition**	**Identification Criteria**	**Occ.**	**References**
End User	The actual user of the Cloud service.	The proposed architectural solution has a visible impact on the service presented to the end user.	6	[26] [36] [39] [59] [132] [141]
Service Provider	The provider of the Cloud service.	The proposed architectural solution explicitly monitors the SLAs fulfillment.	10	[129] [28] [31] [36] [53] [90] [91] [132] [141] [142]
System Architect	The main responsible of the system design [85].	The proposed architectural solution implies an intervention on the business logic of the software system (e.g., invasive monitoring, auto-scaling applications...)	13	[129] [21] [26] [106] [28] [36] [59] [43] [67] [132] [139] [140] [141]
Infrastructure Manager	The responsible for the optimal use of system resources.	The proposed architectural solution implies only an internal reorganization of the computing resources (e.g., consolidation)	12	[22] [30] [49] [50] [53] [80] [90] [91] [112] [115] [140] [142]

Table 4.7: Overview of the identified stakeholders for energy efficiency.

From these numbers, we can observe that Infrastructure Managers and System Architects are, as expected, the most involved by the solutions identified in the primary studies. However, we also notice that End Users are the least involved. This implies that the End User is less aware of the benefits that the solution brings in terms of energy efficiency. Increasing user awareness could instead justify a trade-off between energy efficiency and other crucial quality attributes for the End User (such as performance or usability).

4.6 Threats to Validity

The evidence reported in our work is not immune to validity threats. With respect to the classification done by Wohlin et al. [137], we identify two types of threats, regarding **internal** and **external** validity.

As regards **internal** validity, the main threat concerns the effectiveness of our *search strategy*, as we chose to use only the Google Scholar search engine instead of multiple bibliographic databases. This choice was done after interviewing experts in the field of SLRs in SE. Google Scholar has substantially improved its coverage in the last few years [143] and it is now regarded as an appropriate and comprehensive source [7]. An additional bonus is that this choice simplified the implementation of our search, allowing us to focus more on data extraction and analysis.

Another concern to internal validity regards the *selection process* of the primary studies, as it was carried out by a single researcher. This might have introduced subjective bias in the process. To mitigate those risks, we carefully defined our inclusion/exclusion criteria, to make them as objective as possible. Moreover, the selection process was also carried out in multiple steps (title, abstract and full-text checking) to reduce misinterpretations to a minimum. We adopted a conservative approach, so we are more prone to Type I errors (i.e. false positives) rather than Type II (i.e. false negatives, exclusion of relevant studies).

The main threat to **external** validity is to be found in the data analysis phase. We adopted a *coding* technique to classify the architectural concepts extracted from the primary studies. As coding is a qualitative analysis method, it may be affected by interpretation bias of the individual researcher. To mitigate this risk, the coding process was performed independently by two different researchers, and the resulting lists of codes were merged upon discussion and agreement.

Most of our primary studies present solutions that were never validated in an industrial setting: some of them were validated in academic contexts, through simulation or other similar techniques, while others were not validated at all. Assessing the efficacy of these solutions in tackling energy efficiency issues is out of the scope of this SLR. Nevertheless, we have to consider the lack of validation as a threat to external validity, because it might affect the generalization of our findings. To mitigate this threat, we described the identified architectural concepts in a structured taxonomy, grounded in literature, along with a definition for each concept, hence reducing their specificity to a minimum. This will allow to apply these concepts in real-world case studies, where the impact of our findings on the energy efficiency of software architectures will be properly assessed.

4.7 Conclusions

As data centers are major power consumers, energy efficiency has become a primary issue for cloud service providers. In this context, both the hardware configuration and the software architecture of the cloud computing infrastructure must be carefully designed in order to accommodate power consumption constraints.

In this chapter, we report the results of a systematic literature review that answers RQ 3a, namely: "*Are there software architectural solutions that address energy efficiency aspects?*" Our search resulted in 26 primary studies, mostly published in the last 3 years, each describing a software solution for energy efficiency. Through a coding process, we were able to structure these software solutions in terms of strategies, techniques and components. These concepts provide a common ground for architects to describe, analyze and design energy efficient software solutions.

We identified 3 main strategies: Energy Monitoring, Self-Adaptation and Cloud Federation. It emerged that Self-Adaptation is the most adopted strategy to achieve energy efficiency. However, Cloud Federation will need much more research in the future, due to the diffusion of multi-cloud environments and the need of optimizing the usage of Cloud infrastructures. Regardless of the adopted strategy, fulfilling SLAs constitutes a major concern for software architects. Trade-offs between energy efficiency and other quality attributes are to be further investigated, in order to predict the impact of energy efficient solutions on other service aspects.

We also investigated the stakeholders mentioned in our primary studies. Our results indicate that End-Users are the least involved, which also implies they are less aware of what software does to reduce its energy consumption. Given the massive scale of diffusion of software and services, even a small improvement could contribute greatly. Hence, increasing user awareness could lead to both environmental and economic benefits, as also pointed out by the European Commission in the Horizon 2020 Framework[4]. More research is needed to investigate what to communicate to the user, and how [73].

This chapter gives a comprehensive analysis of the state-of-the-art in energy-efficient cloud software architectures. In the next chapter, we will extract from these results reusable software solutions for designing energy efficient software systems.

[4]`http://ec.europa.eu/research/participants/portal/desktop/en/opportunities/h2020/topics/2360-ee-10-2014.html`

5 A Catalog of Green Architectural Tactics for the Cloud

In the previous chapter, we analyzed the literature and elicited a set of techniques for addressing energy efficiency in cloud-based software architectures. In this chapter we codify these techniques in the form of Green Architectural Tactics. These tactics will help architects extend their design reasoning towards energy efficiency and to apply reusable solutions for greener software. This contribution answers RQ 3b.

5.1 Introduction

As the adoption of cloud computing technologies continues to grow, the need for energy-efficient solutions becomes evident. Cloud-based software holds great potential for energy efficiency: a recent study [82] showed that migrating all business applications in the U.S. to the cloud could reduce their energy footprint by 87%. A previous work [45] started to analyze the cost and energy benefits of data migration to the cloud.

However, this transition to the cloud is not an easy task. Cloud-based software must be appropriately designed to address energy efficiency, which is typically not the case for traditional business applications. If these applications are abruptly migrated, it is highly likely that the resulting energy waste would significantly outweigh the expected benefits.

In the previous chapter, we presented the results a Systematic Literature Review (SLR) on software architectural solutions for cloud-based software that addressed energy efficiency-related issues. The SLR identified a number of recurring techniques that were potentially reusable in other solutions. In this chapter, we codify these techniques in the form of Green Architectural Tactics. These tactics can be adopted by software architects and developers during the design and development of cloud-based software systems or the refactoring of existing business applications for cloud migration. This contribution will support decision-making when dealing with energy efficiency aspects of cloud-based software architectures.

The chapter is structured as follows: in Section 5.2 we present similar approaches and efforts that address energy efficiency as an architectural concern. Section 5.3 introduces energy efficiency as a quality attribute. In Section 5.4 the Green Architectural Tactics are presented and described with application examples extracted from the literature. Section 5.5 discusses the architectural implications of energy efficiency. Finally, Section 5.6 presents our strategy for evaluating the impact of the Tactics and Section 5.7 concludes the chapter.

5.2 Related Work

While a steadily growing scientific body is being built on green software engineering [20], most research focuses on estimating or measuring power consumption at the system- or source-code level, without suggesting ways to actually develop energy-aware software (e.g. [45, 82]). Very little research has been carried out in studying energy efficiency at the software architecture level, neither in general nor for cloud-based software. Some preliminary investigations go back to the work of Rangaraj & Bahsoon [103], who used market-based economics theory to define a framework for optimizing power consumption in energy-unaware software architectures at runtime. Bahsoon then planned to apply the same approach to cloud architectures [6]. In [113], Seo et al. come closer to our objective by defining a framework that estimates the energy consumption of three distributed system architectural styles. Their goal is to evaluate the most appropriate architectural style *before* implementation. Te Brinke et al.[124] propose a design method to extend modules with energy optimizers. Although Rangaraj and Bahsoon agree with us in considering architecture the right abstraction level for addressing energy-related concerns [103], no work so far has provided support for architects to actually design software architectures that address energy efficiency upfront.

For software system architectures the story is not that different. For example, energy efficiency in mobile computing is a widely addressed topic because of battery limitations of mobile devices [33]. Cyber-foraging, a form of mobile cloud computing in which mobile devices offload expensive computation to more powerful servers in the cloud, is a common strategy for saving battery power on mobile devices [109]. However, it is not uncommon for literature on cyber-foraging to refer to the cloud as having infinite resources; which means that no reusable cyber-foraging strategies have been defined yet for architecting energy-aware software systems that address the energy efficiency of the system as a whole.

That being said, there are some existing tools that can be used to implement tactics for energy efficiency in Cloud-based software. Amazon Web Services (AWS), for example, provides an Auto Scaling[1] feature that scales the capacity of VM instances (EC2, Elastic Compute Cloud) elastically depending on user-defined conditions. In addition, Amazon also provides CloudWatch[2], a Web Service that monitors several metrics of the EC2 instances that can be used to trigger Scaling operations. These tools can be used to implement either Energy Monitoring or Self-Adaptation tactics for energy efficiency, later described in this chapter.

5.3 Energy Efficiency as a Quality Attribute

According to Bass et al. [12], energy efficiency is to be regarded as a "system" quality attribute because it is the result of an indirect action of software. However, Bass et al. also argue that the line between "software" and "system" quality attributes is very thin. In the end, even if energy is ultimately consumed by hardware, it is software that determines hardware behavior. In order to provide a clear representation of energy efficiency as a quality attribute, we follow the approach adopted by Bass et al. [12] and characterize energy efficiency through *quality attribute scenarios*. Each scenario is described in terms of six characteristics:

- *Stimulus.* An event that motivates an action concerning energy efficiency.
- *Source of Stimulus.* The entity that triggered the event.
- *Environment.* The set of circumstances under which the scenario takes place.
- *Artifact.* The element of the system that is stimulated by the event.
- *Response.* The action to be performed in response to the event.
- *Response Measure.* The metric that determines if the response is satisfactory.

We grouped our Green Architectural Tactics in three categories and formulated a scenario for each category (see Tables 5.1 and 5.2). In all the scenarios, the *response measure* is energy consumption values. In the following section, we describe the identified scenarios for each category, as well as the elicited Green Architectural Tactics.

[1]`http://aws.amazon.com/autoscaling/`
[2]`http://aws.amazon.com/cloudwatch/`

Table 5.1: Quality Attribute scenarios for Energy Efficiency

	Energy Efficiency Scenarios		
Category	*Energy Monitoring*	*Self-Adaptation*	*Cloud Federation*
Stimulus	Request for energy consumption information	Energy consumption alert	Energy consumption alert
Source of Stimulus	Administrator	Energy Monitor	Energy Monitor
Environment	Normal operation	Runtime	Multi-cloud
Artifact	Energy Monitor	Hypervisor	Orchestrator
Response	The Energy Monitor presents the detailed energy consumption information for the data center.	The Hypervisor consolidates the VMs on the less-active servers and then shuts down the idle servers.	The Orchestrator swaps the most energy-consuming services with less energy-consuming services.
Response Measure	Energy consumption values	Energy consumption values	Energy consumption values

5.4 Green Architectural Tactics

In the previous chapter we identified a set of recurring design solutions, described in the literature, to achieve energy efficiency in cloud-based software architectures. In this work, we codified these solutions as *tactics* – that is, “design decisions that influence the achievement of a quality attribute response” [12]. Each tactic is described in terms of:

- *Motivation:* rationale behind the tactic.
- *Description:* components introduced by the tactic and their roles.
- *Constraints:* necessary conditions for applying the tactic in an existing software architecture.
- *Example:* previous application of the tactic.
- *Dependencies:* whether the tactic requires other tactics to be applied.

In the following, we describe the identified scenarios for each category, as well as an example of a Green Architectural Tactic.

5.4.1 Energy Monitoring

A typical scenario for energy efficiency that involves Energy Monitoring is the following: the system administrator of a cloud-based system wants to know the energy consumption of its infrastructure during operations. The Energy Monitor gathers the energy consumption information and presents it to the administrator.

The tactics in this category are targeted at monitoring the energy consumption of the cloud infrastructure. These tactics are often combined with tactics from other categories; Self-Adaptation in particular because information from monitoring components is typically used to trigger adaptive mechanisms.

Table 5.2: Overview of Energy Efficiency Tactics

Category	**Tactics**
Energy Monitoring	Metering Static Classification Modeling
Self-Adaptation	Scaling Down Consolidation Workload Scheduling
Cloud Federation	Energy Brokering Service-Adaptation

Metering

Motivation. Instrumenting a data center with power metering devices is becoming common practice[3]. The market is flooded with many different models of power meters with enhanced capabilities (e.g., wireless communications, high sampling frequencies, data analysis features). Many devices come with built-in sensors and tools to monitor power consumption in real-time. The Metering tactic enables to effectively use the information provided by these devices.

Description. The Metering tactic consists of collecting power metering information from the hardware through dedicated software components called Energy Collectors. Collectors are usually in a many-to-many relationship with physical power meters. These Collectors share information via an Energy Communication Bus (ECB) that provides a common interface for energy information. In addition, the energy consumption information is stored in a dedicated Energy Database that can have different levels of granularity. Finally, a GUI component called an Energy Dashboard provides graphical representations of energy information along with useful reporting for both cloud service providers and customers.

Constraints. The main limitation of this tactic is the need for a physical metering infrastructure, which can be costly in the case of large data centers. In addition, the granularity of the information gathered and shared by the metering process has to be tuned accordingly in order to avoid information overload.

Example. An example of how to apply the Metering tactic is shown in the CompatibleOne project [21]. CompatibleOne is a cloud resource management software that allows the creation of hybrid cloud platforms through the aggregation of services from different cloud providers. In this context, an Energy Monitoring framework was developed to monitor the energy consumption of each cloud provider participating in the platform for subsequent energy billing and environmental impact evaluation. Several power meters and probes are supported by the framework. As shown in Figure 5.1a, starting from the hardware layer at the bottom of the figure, the power consumption data flows from the physical resources through the probes. A Collecting Daemon, of type Energy Collector, retrieves the data through the GetValue interface of the probes. This data is then converted into XML format and stored in a BerkeleyDB database (Energy Database). Another software component, a DatabaseDaemon that acts as an Energy Communication Bus, provides access to the database to HTML and PHP front-ends (Energy Dashboard) through an Open Cloud Computing Interface, a standard set of specifications for cloud computing providers. Finally, the front-ends present the information to the system administrators.

[3]`http://bit.ly/1yq1jrq`, last visited on October 1st, 2013.

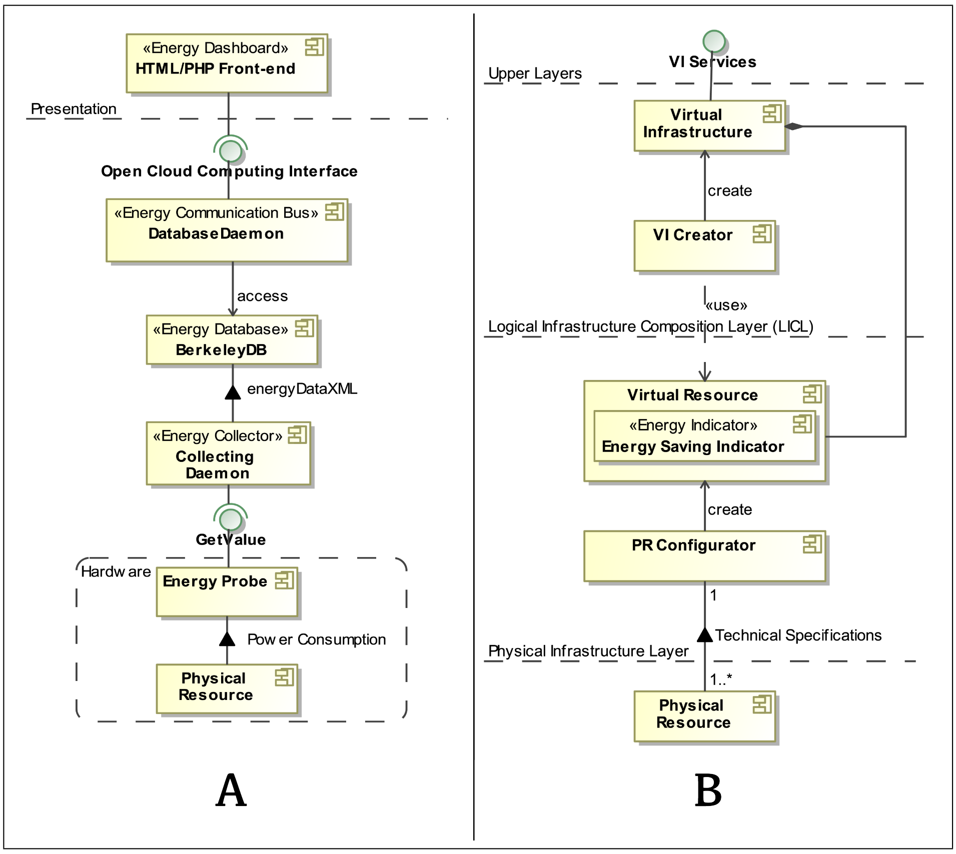


Figure 5.1: A. Example of the Metering tactic. B. Example of the Static Classification tactic.

Static Classification

Motivation. A cloud infrastructure is typically composed of many heterogeneous IT devices. Direct energy consumption monitoring of each one of these devices might be infeasible because the physical machines might be external to the organization of the cloud software provider. The Static Classification tactic provides a solution to estimate the power consumption of the infrastructure when metering information is unavailable.
Description. This tactic consists of classifying the different resources in terms of energy efficiency through the use of Energy Indicators. This classification is static, i.e., not based on on-line, real-time information, but rather on technical

specifications and characteristics of the devices themselves. To some extent, the Energy Indicators share an analogy with the Energy Labels[4] designed by the EU to classify the energy efficiency of appliances.
Constraints. Unfortunately, hardware vendors not always disclose energy consumption specifications of their products. In addition, this tactic is not applicable to any operation that requires an on-line analysis of software behavior on a fixed physical platform.
Example. An example of this tactic can be seen in the GEYSERS EU project [129]. The context is a multi-layered software architecture for dynamic cloud service provisioning in which the Physical Infrastructure Layer (PIL) is decoupled from the Logical Infrastructure Composition Layer (LICL). One of the goals of the project is the selection of the "greenest" physical resources, based on Static Classification, to create energy-efficient Virtual Infrastructures (VIs). The example is described in the component diagram in Figure 5.1b: the PR Configurator takes as input the technical specifications of physical resources and assigns "Energy Saving Indicators" (ESIs), of type Energy Indicator, to created Virtual Resources. Subsequently, the VI Creator component in the LICL composes Virtual Resources into VIs, using the ESIs to prioritize the selection of the resources. Finally, the LICL exposes the services provided by the VIs to the upper layers of the architecture.

Modeling

Motivation. In order to implement self-adaptive mechanisms it is necessary to have near-real-time energy consumption information. This enables the modification of software behavior according to how much energy the system is actually consuming. When metering systems are unavailable, the Modeling tactic is a viable option.
Description. The Modeling tactic enables a dynamic estimation of power consumption values through predictive Energy Models. These Models are embedded in Energy Indicators, similar to those in the Static Classification tactic. However, these Energy Indicators do not statically classify physical resources, but rather provide a dynamic estimation of the power consumption of the software components. Typically, Energy Models are built through regression analysis based on software runtime metrics, i.e. resource usage (CPU, disk, memory) [104].

Constraints. The limitation of this tactic lies in the accuracy of the software Energy Models. To date, many models and tools are available to estimate software energy consumption but their accuracy varies greatly based on the selected hardware platform. In addition, not all hardware resources are good predictors of energy consumption; identifying the best predictors is still an issue for researchers

[4]`http://www.newenergylabel.com/index.php`, last accessed on September 18, 2013.

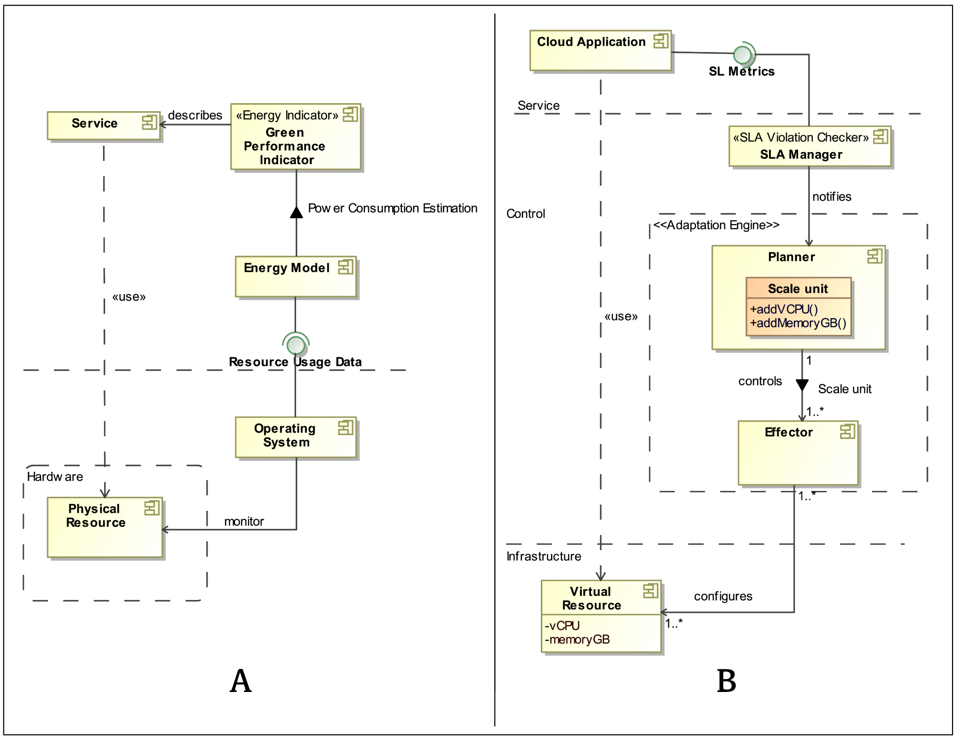


Figure 5.2: A. Example of the Modeling tactic. B. Example of the Scaling Down tactic.

in the field.
Example. A prototype showing the application of this tactic is provided by de Oliveira et al. [28]. The context is a Service-Oriented Architecture (SOA) applied to a cloud infrastructure. As shown in Figure 5.2a, for each service of the SOA, the Operating System of each physical node provides service-related Resource Usage Data (in the case of the example, CPU, memory and disk [28]). A linear Energy Model retrieves this data and estimates the power consumption impact of each service. The estimation is modeled into a Green Performance Indicator (GPI), of type Energy Indicator. Each GPI describes a service in terms of energy efficiency.

5.4.2 Self-Adaptation

The Self-Adaptation scenario for energy efficiency starts from the Energy Monitor that reports an alert of excessive energy consumption while the system is not fully loaded. In response, the Cloud Hypervisor (i.e. the Virtual Machine Monitor [99]) migrates some of the VMs to less-loaded servers so that it can shut down the resulting idle servers.

Tactics in this category implement mechanisms that modify runtime software configurations for the specific purpose of lowering energy consumption. In cloud-based environments Self-Adaptation mostly concerns the configuration, deployment, and workload of Virtual Machines (VMs).

Scaling Down

Motivation. One of the key features of cloud computing is the ability to provide resources on demand. When more resources are needed to satisfy incoming requests, the cloud infrastructure allocates more physical resources to VMs (*scaling up* or *vertical scaling*). However, the opposite mechanism should also be in place: when a decrease in demand occurs VMs must be appropriately scaled *down* in order to avoid energy waste. The Scaling Down tactic describes how to design this mechanism.
Description. An important component of this tactic is the Scale Unit, i.e., a pre-defined "block of IT resources" [106] explicitly modeled as a software component. Modeling Scale Units is useful for planning the scaling operations because it defines a finite number of configurations for the VMs. Thus, it is possible to associate each configuration with a particular level of demand or system load. The Adaptation Engine is the component that performs the Scaling operation; this role is typically played by the Hypervisor. Another key component is the SLA Violation Checker. During the Scaling operation the fulfillment of established service-level objectives must be ensured at all times. This component performs the needed checks and accordingly allows or disallows the Adaptation Engine to perform the Scaling.
Constraints. Scaling is a complex operation that requires careful planning and continuous monitoring. The main challenge is determining the right amount of resources that define a Scale Unit. This implies the prediction of expected levels of demand, which is not an easy task especially in large-scale cloud service provisioning.
Example. A possible implementation of the Scaling Down tactic is provided by Xu et al. [141]. As illustrated in Figure 5.2b, each Virtual Resource is configured by the Adaptation Engine, realized by the Effector and the Scheme Planner. The Effector actively executes the scaling of the Virtual Resources, by a number of Scale Units determined by the Scheme Planner that evaluates the current VM

configuration considering the requirements of the system. In this example, Scale Units are modeled in terms of assigned virtual processors (vCPUs) and memory size (memoryGB). The Virtual Resources are used by the Cloud Application that exposes its service-level metrics via a REST API (SL Metrics). The CApp SLA Manager, of type SLA Violation Checker, monitors those metrics to ensure that service-level objectives are met. If necessary, the CApp SLA Manager issues a notification to the Scheme Planner to scale up the Virtual Resources again.
Dependencies. The Scaling Down tactic requires some sort of Energy Monitoring tactic for the Adaptation Engine to decide whether or not to perform scaling operations. In the previous example a Metering tactic is implemented by Sensors (Energy Collectors) that collect energy-related metrics; a Monitoring Center (Energy Monitor) that records, filters and audits the data provided by the sensors; and a Knowledge Base (Energy Database) where energy consumption information is stored.

Consolidation

Motivation. As mentioned earlier, on-demand resource provisioning is an important feature of cloud-based environments. Adding resources to a single VM may not always be the best option. For example in cloud application server provisioning[5] creating new VM instances may provide additional flexibility and help to perform load balancing among servers. This is called *horizontal* scaling (or scaling *out*). This operation, however, may easily lead to inefficient usage of physical resources if the density of VMs across the physical servers is not accurately managed in low-request phases. Indeed, the Consolidation tactic concentrates the VM instances on the minimum number of servers needed. Powering down the unused servers will evidently increase the energy efficiency of the cloud-based software.
Description. The main component of the Consolidation tactic is the VM Allocator, the software component responsible for live VM migration. This component can be (a part of) the Hypervisor, as in the Adaptation Engine in the Scaling tactic. The SLA Violation Checker is needed as well to check the fulfillment of service-level objectives after VM migrations.
Constraints. Consolidation must take place at runtime. This means that VMs must be represented in a format that allows them to be seamlessly migrated from one location to another, along with their context, workload, and metadata. This may introduce high network traffic and security risks.
Example. Dupont et al. [31] provide a sample implementation of the Consolidation tactic, depicted in Figure 5.3a. The Power Calculator, of type Energy

[5]`http://www.enterprisenetworkingplanet.com/netos/article.php/3753836/Practical-VM-Architecture-How-Do-You-Scale.htm`, last visited on Oct. 2013

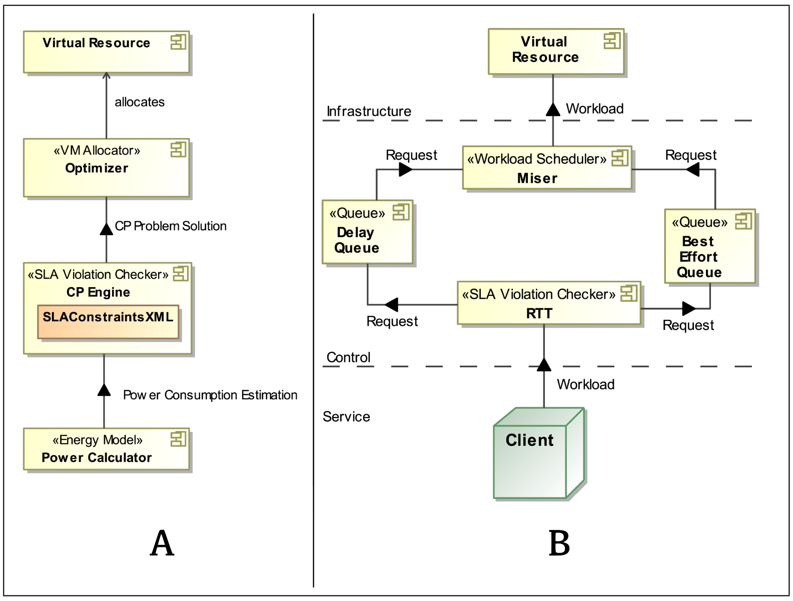


Figure 5.3: A. Example of the Consolidation tactic. B. Example of the Workload Scheduling tactic.

Model, provides a power consumption estimation to the CP Engine, of type SLA Violation Checker. The CP Engine formulates a constraint programming problem using the constraints extracted from the SLAs in XML format (SLA-ConstraintsXML). The CP engine then solves the problem and the Optimizer (of type VM Allocator) produces a VM allocation scheme by applying the solution to the Virtual Resources.
Dependencies. The presence of the Power Calculator indicates a dependency on the Modeling tactic: as shown in Figure 5.3a, the Power Calculator is an instance of an Energy Model.

Workload Scheduling

Motivation. The property of adapting to workload changes by provisioning and de-provisioning resources is called *elasticity* [51] and it is commonly regarded as a defining property of cloud environments. Elasticity has, of course, a direct connection with energy efficiency: the more closely resource provisioning matches demand, the more energy efficient the infrastructure is. The Scaling Down tactic allows to adapt resource provisioning, while the Workload Scheduling tactic is

meant to prioritize and assign the load to the different virtual resources in order to match the demand.
Description. In this tactic, a Workload Scheduler is a software component that is able to dispatch workloads to VMs. The Scheduler normally uses one or more Queues to arrange the workloads. Queues can be differentiated in terms of priority levels, QoS requirements or deadlines. The SLA Violation Checker ensures that all service-level objectives are met.
Constraints. Workload scheduling is a well-known practice in software systems that is widely studied in operating systems theory. However, workload scheduling has specific challenges in cloud-based environments. First, when modeling workloads it is necessary to select the appropriate workload granularity. For example, a workload can divided per application, VM, or pool of VMs. In addition, efficient workload prediction in cloud environments is difficult to achieve because of the high variability of demand.
Example. Lu et al. [80] provide an example of Workload Scheduling for cloud storage services. In their solution, shown in Figure 5.3, when a Client node submits a Workload to the service, the RTT algorithm, of type SLA Violation Checker, decomposes the Workload into Requests, to be assigned to different Queues, according the deadline of each Request. In the example, two Queues are present: a Delay Queue that has a guaranteed response time and a Best Effort Queue that has no time constraints. The Miser algorithm, of type Workload Scheduler, is used to recombine Workloads and dispatch them to Virtual Resources.

5.4.3 Cloud Federation

The Cloud Federation scenario for energy efficiency is the following: the Energy Monitor notifies about excessive energy consumption arising from a service, which is a composition of multiple cloud services. The Service Orchestrator then tries to swap some services in the service composition by searching in a Green Service Directory for iso-functional services that consume less energy than those currently being used.

A cloud federation is a multi-cloud environment that can be defined as "[*a platform that*] comprises services from different providers aggregated in a single pool" [68]. Cloud Federation tactics allow cloud-based software systems to "lease" or "negotiate" cloud services from multiple providers based on energy consumption information.

Energy Brokering

Motivation. Service discoverability is one of the key principles of service orientation [32]. To enable cloud service composition in multi-cloud environments, the same principle applies. The Energy Brokering tactic makes energy information

about services an additional parameter for service discovery and selection.
Description. This tactic is realized by means of two components: an Energy Broker and a Green Service Directory (GSD). The Energy Broker is a service that enables access to energy-efficient services. It receives requests for cloud services that perform a specific task and returns a pointer to the most energy-efficient service available in the multi-cloud that can perform the requested task. To do so, Energy Brokers make use of a GSD, which is a repository where all the cloud providers in the multi-cloud store the energy information of the services they provide.
Constraints. This tactic does not specify where the Energy Broker and the GSD should be hosted. However, for trust reasons, they should not be hosted by any cloud service provider participating in the federation.
Example. Garg et al. [39] propose a framework called Green Cloud Architecture that serves as an example for the Energy Brokering tactic. We model this example by means of a communication diagram (see Figure 5.4a) as a specific behavioral interaction is suggested. In the Figure, a Green Broker, instance of an Energy Broker, accepts requests for cloud services. The Broker queries the Green Offer Directory (GOD) that lists all green cloud services available. The GOD returns a list of services able to fulfill the request. The Broker then queries the Carbon Emission Directory (CED) to discover the specific energy efficiency information for each service. The combination of the CED and the GOD realize the GSD of the tactic. Finally, the Broker fulfills the request with the most energy-efficient service available.
Dependencies. The Green Service Directory has to characterize each service with its energy information. This requires Energy Indicators for each service, either static or dynamic, which suggests a dependency with either the Static Classification or the Modeling tactic, respectively.

Service-Adaptation

Motivation. The main benefit of the Cloud Federation paradigm is the possibility to select services among different providers. The Energy Brokering tactic provides the energy information for services. This enables cloud-based software systems to discover services that are more energy-efficient than those currently in use. The Service-Adaptation tactic describes how Cloud platforms should switch to these more energy-efficient services.
Description. Two components realize the Service-Adaptation tactic. The first component is the Energy Orchestrator that communicates with the Energy Broker to discover energy-efficient services that fulfill a certain task and eventually performs the registration of those services with the system. This operation has to be authorized by the second component, the SLA Violation Checker, which ensures that the new services meet the service-level objectives required by the

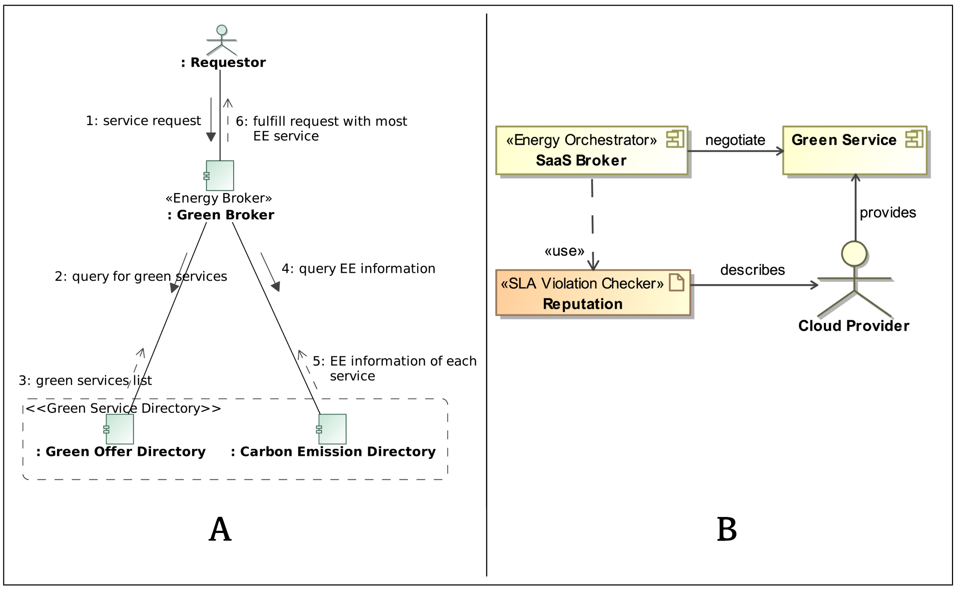


Figure 5.4: A. Example of the Energy Brokering tactic. B. Example of the Service-Adaptation tactic.

system. This component is similar to its analog in the Self-Adaptation tactics, but instead of checking the SLOs that *internal* services have to fulfill, it checks if *external* services meet the SLOs required by the system.
Constraints. The Service-Adaptation tactic assumes centralized cloud service orchestration. This creates some disadvantages in terms of flexibility because it concentrates all service orchestration logic in a single point.
Example. Villegas et al. [132] illustrate an example of the Service-Adaptation tactic in a federated cloud architecture. In their view, the Service-Adaptation is performed at the Software-as-a-Service (SaaS) layer: whenever a service request to the federated cloud cannot be fulfilled with the required service level or is too costly in terms of energy, it is forwarded to another federated cloud provider. As shown in Figure 5.4b, the SaaS Broker, of type Energy Orchestrator, negotiates the usage of a Green Service with other cloud providers. The Reputation of the cloud provider (of type SLA Violation Checker) determines if the provider meets the required service-level objectives. The Reputation is based on the SLA violation rate of the provider.
Dependencies. As implied by the tactic description, Service-Adaptation depends on the Energy Brokering tactic in order to retrieve the energy information of

services.

5.5 Discussion

The Green Architectural Tactics presented in this work were explicitly formulated with reusability in mind. For this reason, we kept to a minimum the constraints that a tactic may impose on the general software architecture. When necessary, we made them explicit. For example, the Service-Adaptation tactic assumes the presence of a service orchestration mechanism; most of the Energy Monitoring tactics introduce a centralized Energy Database; Self-Adaptation tactics assume a high degree of decoupling between the virtual and the physical infrastructure. If these tactics are meant to be applied to an existing cloud-based system, software architects should consider whether these assumptions are compatible with the current architecture.

An alternate top-down design approach could be to describe our Tactics using a higher-level pattern language. An example might be the MAPE-K pattern [61]: Energy Monitoring Tactics can be adopted to implement the Monitoring and Analysis function, and Self-Adaptation can be adopted for Planning and Execution.

However, it is important to note that Green Architectural Tactics cannot generally be adopted in isolation: when introducing them in a software architecture, they might require other tactics to be adopted as well. In the previous section, we made such dependencies explicit. In short, we found that Energy Monitoring tactics are required whenever Scaling Down, Consolidation and Energy Brokering are adopted. In addition, Service Adaptation requires Energy Brokering to function properly. It is relevant to point out that the occurrence of a combination of tactics does not always imply a dependency. For example, the dependencies that emerged from our SLR have identified, in a total of 26 primary studies, the following combinations (see Chapter 4):

- Energy Monitoring and Self-Adaptation tactics, in 8 cases.
- Self Adaptation and Cloud Federation tactics, in one case.
- Energy Monitoring, Self-Adaptation and Cloud Federation tactics, in 2 cases.

This evidence suggest a deeper relationship between the tactics that we will further explore in our future research. Furthermore, our tactics introduce tradeoffs between energy efficiency and other quality attributes, summarized in Table 5.3. Along with the scenarios provided in Section 5.4, this initial trade-off analysis contributes to the identification of energy efficiency as a quality attribute. It is still under discussion to what extent energy efficiency and other sub-characteristics of environmental sustainability might influence traditional quality requirements.

Table 5.3: Energy efficiency tradeoffs introduced by Green Architectural Tactics

Tactic	*Quality Attribute*	*Rationale*
Scaling Down	Performance	Scaling down VMs may result in lower performance in case of unanticipated demand spikes.
Consolidation	Availability	During VM migration some services may not be available.
	Security	Live VM migration over the network requires to transfer application code, metadata and workloads, making them vulnerable to attacks.
Modeling	Modifiability	Energy Connectors are component-specific and therefore must be reimplemented if the architecture changes.
Service-Adaptation	Flexibility	The orchestrator concentrates all service composition logic in a single node.
Workload Scheduling	Performance	If workload prediction fails deadlines might be missed.

5.6 Next Steps: Tactics Evaluation

Because tactics are elicited from specific implementations, they do not come with generalizable measures of the potential energy savings that they provide. As reported in Chapter 4, most of the primary studies included a validation phase, performed in either an industrial or academic setting. As part of our future work, we plan to conduct research activities to provide an estimation of the impact of the adoption of the Tactics on energy consumption.

A first step will be an industrial survey among experts of the field (i.e. software architects) to have a first evaluation and prioritization of the tactics in terms of their potential impact. We already contacted a number of interested participants through our network in the Green IT Amsterdam[6] and in the EFRO MRA Cluster Green Software project[7] consortia.

Secondly, after this exploratory study, we plan to set up empirical experiments aimed at quantitatively assessing the impact of the Tactics. The experiments will be carried out on instrumented environments where we will monitor the execution of Cloud-based software applications implementing our Tactics. Meanwhile, we will gather fine-grained energy consumption data that will allow us to evaluate the energy savings gained through the Tactics implementation. For this research, we

[6]`http://www.greenitamsterdam.nl/`
[7]`http://www.clustergreensoftware.nl/`

will collaborate with Cloud service providers based in Amsterdam for providing case studies and with the Hogeschool van Amsterdam (HvA) for their expertise in hardware instrumentation and measurement. We will also make use of our cluster computing resources at the VU University Amsterdam as a testbed for the experimentation.

5.7 Conclusions

In this chapter, we describe energy efficiency as a software quality attribute and analyze its architectural impact in terms of assumptions and trade-offs. This chapter answers RQ 3b, namely "*How can architectural solutions for energy efficiency be made reusable?*": we provided a set of reusable design solutions, codified as *tactics*, to support the design and development of cloud-based energy efficient software. In order to help their understanding and adoption, each of our Green Architectural Tactics is presented with an example of its application extracted from the literature.

Together with Chapter 4, in this part of the thesis we answer RQ 3, namely "How can software architectural solutions realize energy efficiency?" By eliciting existing solutions and generalizing them into a catalog of reusable tactics, we show how energy efficiency can be realized through software architecture. In the final part of this thesis, we will provide a conceptual framework to encapsulate the experience gathered so far into strategies for energy-efficient software.

6 A Conceptual Framework for Energy-Efficient Software Engineering

The pivotal role of software in energy consumption is now supported by sound empirical data collected through a series of experiments on different hardware platforms. Although the actual figures may vary depending on the specific platform, the impact of software over energy consumption is definitely relevant. Addressing this impact requires a change of mindset from the software engineering community. In this chapter, we present a conceptual framework that provides a unifying view on the strategies, models and tools presented so far to engineer energy-efficient software. This contribution answers RQ 4.

6.1 Introduction

The theoretical software power models presented in Chapter 2 give developers a way to elaborate a strategy by analyzing the causes of energy consumption. Moreover, power models and measurement techniques are needed to validate the efficacy of the formulated strategies by measuring their impact. The empirical evidence presented in this thesis represents a starting point for such strategies: for example, in Chapter 3 we validated two practices for energy-efficient software development. These guidelines can be embedded in a strategy to refactor existing software applications and increase their energy efficiency. In Chapter 5 we presented three strategies to address energy efficiency aspects in Cloud software applications. Some of these strategies can be applied in a more general software engineering process to design energy-efficient software applications from scratch.

In this chapter, we provide a conceptual framework to support energy-efficient software engineering. Our framework is based on the results obtained to far in this thesis. It makes use of already existing tools and techniques, and we also provide examples of useful measurements and metrics to ease its adoption.

This chapter is structured as follows: in Section 6.2 we provide reflection upon the results of the empirical experimentation we performed in software energy efficiency. In Section 6.3 we introduce our conceptual framework. In Section 6.4 we present a more general overview of the stakeholders for software energy efficiency. In Section 6.5 we describe the strategies included in our framework in detail. Finally, Section 6.6 concludes the chapter.

6.2 Reflection on Empirical Evidence

In Chapter 2, we already discussed the modeling approaches for software energy consumption: they can be "*white-box*" e.g. based on code-level or instruction-level metrics, or' "*black-box*" e.g. based on runtime measurements, such as usage ratios of system resources (CPU, RAM, etc.).

From our experience, the effectiveness of the chosen predictors varies greatly with respect to the considered hardware configuration. In embedded systems, for example, we have observed that code-level constructs may have an observable impact over power consumption only in some cases [131]. However, as the system architecture becomes more complex, these models appear to be too fine-grained to describe the effect of software over power consumption. In these cases, a resource-usage based model might be more meaningful: our empirical studies in Chapter 2 and 3 have successfully proven the correlation between indicators of hardware resources and the power consumption of computer systems. As a matter of fact, most of the software power profiling tools commercially available are based upon these types of models.

Choosing the appropriate resources (or, more precisely, resource usage metrics) as predictors is the key to build an accurate resource-based power consumption model. Typically, CPU is the most important component to monitor: this is why, especially on more advanced mobile systems such as smartphones, most tools focus on the CPU usage as a predictor for software power consumption. However, our experiments have proven that other metrics, such as memory usage and, more importantly, I/O operations, have a significant correlation with power consumption. Moreover, in some usage scenarios, software applications may require the activation of high power-consuming peripherals (e.g. GPS modules, 3G and WiFi antennas) that significantly modify the consumption profile of the device. In Figure 6.1 we can see an example showing typical power consumption of a mobile device in different usage scenarios [2] .

This suggests two considerations: first, these resources cannot be ignored by models and must be explicitly measured. Secondly, software developers need to be aware that decreasing the computational complexity of software applications is not enough to develop an energy-efficient application.

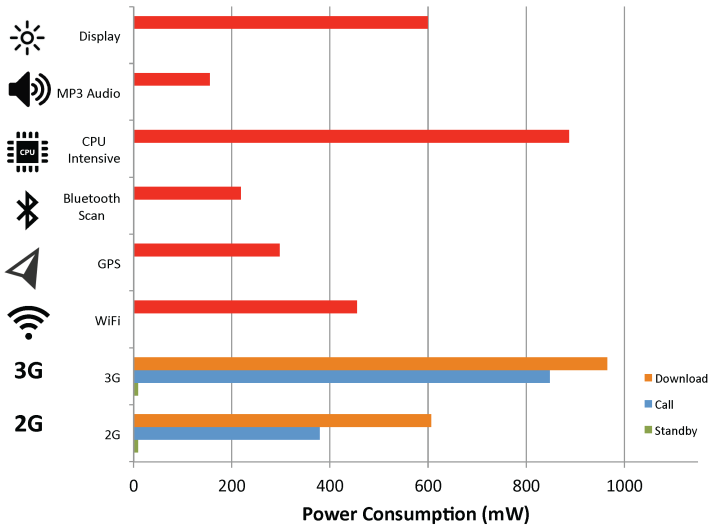


Figure 6.1: Power Consumption of a Mobile Device in Different Usage Scenarios (data from [2])

6.3 Conceptual Framework

Our framework to engineer energy-efficient software is presented in Figure 6.2.

The framework includes three main strategies: *Energy Monitoring, Refactoring and Self-adaptation.* The *Refactoring* strategy is described in detail in Section 6.5.2. It is focused on minimizing software instructions and code patterns that may cause higher energy usage. *Energy Monitoring* and *Self-Adaptation* were earlier introduced specifically for Cloud-based software in Chapter 4 and 5[1]. Here we present them in a more general formulation: the *Self-Adaptation* strategy aims at creating an energy-aware application that is able to choose among various configurations, here called "energy profiles" (see Section 6.5.3), depending on the

[1]The third strategy we introduced, *Cloud Federation*, is Cloud-specific as it is based on multiple Cloud services, from different providers, interacting with each other. Hence, it is not suitable to be included in our framework.

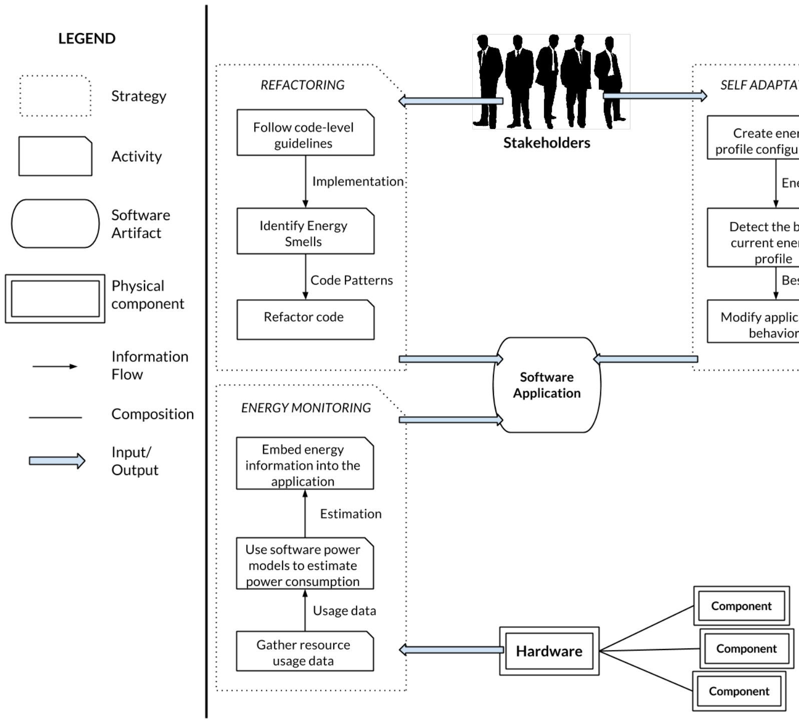


Figure 6.2: Framework for Energy-Efficient Software Strategies

scenario and the execution context. The *Energy Monitoring* strategy (see Section 6.5.1) aims at providing feedback on software energy consumption through modeling and profiling.

As shown from Figure 6.2, the input/output flow of the strategies can be both bottom-up and top-down. From the bottom, hardware information is injected into software applications to create energy awareness. From the top, the stakeholders who are interested in or affected by energy efficiency issues (see Section 6.4) trigger the need for energy-efficient software. This serves as a top-level input, to determine which operational decisions for software energy efficiency are to be taken.

The strategies are not meant to be mutually exclusive: as seen in Chapter 4, they can be applied together in various combinations. In addition, other technological, human or process strategies can be plugged in, provided that their impact in terms of energy consumption is verifiable through measurements or estimations.

6.4 Stakeholders

In Chapter 4 we presented a first overview of the stakeholders that we identified during our SLR on Cloud software architectures. For the purpose of that study, we focused on the energy efficiency of Cloud services during its usage and provisioning. In this section we provide an extended overview, from a wider perspective on software systems. We refer to the standard definition of a *stakeholder* [55], namely: *"individual, team, organization, or classes thereof, having an interest in a system"*. Hence, we define stakeholders for software energy efficiency as *individuals, teams, organizations, or classes thereof, having an interest in improving the energy efficiency of a software system"*.

A more generic list of stakeholders for software sustainability is also provided by Penzenstadler et al. [95]. The stakeholders identified in their work take into account multiple dimensions of sustainability (individual, social, economic, environmental, technical).

End Users are mainly interested in software energy efficiency for usability reasons: the proliferation of mobile, battery-powered devices made users aware that a long-lasting battery also depends on software activities. In turn, software energy efficiency is becoming an important parameter for choosing the right application to perform a task. Ultimately, this will push the market to produce more energy-efficient software and end-users are the main drivers of this process.

Software Developers play a very important role. Their interest is to constantly monitor and optimize the energy efficiency of their applications, through the support of appropriate automated tools. Their feedback is crucial in order to establish practical guidelines to write energy efficient software.

Software Engineers are responsible of enabling and investigating this innovation. The software engineering community has already shown its interest in the energy footprint of software: the International Workshops on Green and Sustainable Software (GREENS) [71, 73] are a prominent example. The involvement of software engineers is crucial in order to embed sustainability in development processes.

System Architects are the main responsible of the system design and they must have a "broad, global, whole-system view" [85]. The energy efficiency of the system as a whole is nowadays a primary requirement in many contexts (e.g. High-Performance Computing, mobile devices, embedded systems). Addressing such a requirement implies to take significant design decisions involving multiple components and different abstraction layers of the system. Hence, architects are to be considered stakeholders as well.

Infrastructure Managers are responsible for the optimal use of system resources. In IT infrastructures, energy is one of the most expensive assets. Increasing the energy efficiency of software means being able to perform more tasks using the same amount of energy, thus infrastructure managers definitely have an interest in it.

Service Providers are interested in energy efficiency for multiple reasons. There's an economic reason, as energy is one of the main voices in the Total Cost of Ownership of large-scale service provisioning infrastructures. Sector leaders, such as Google[2] and Microsoft[3], already undertook initiatives to reduce the energy footprint of their software services. Moreover, as the general awareness towards the energy impact of software increases, service consumers will ask for explicit levels of energy efficiency in their service agreements. It is expected that sooner or later, all providers will include energy efficiency as one of the parameters to evaluate the quality of their service.

6.5 Strategies for Energy-Efficient Software

6.5.1 Energy Monitoring: use software energy models to drive improvements

The first strategy we present is Energy Monitoring: its aim is to provide feedback on the energy consumption of software applications, to identify opportunities for energy optimization and/or to assess the energy savings gained by applying other strategies. The bottom part of Figure 6.2 shows the information flow coming from the hardware level: resource usage data, e.g. memory accesses, I/O usage, CPU usage, is collected from the hardware. This information is used as input for software energy models, that analyze applications during execution and provide on-line energy consumption estimations with different granularity. Example of already available profiling tools that make use of energy models are: *Joulemeter* [83], *ARO* [4], *Power TOP* [54] and *PowerTutor* [41].

The Energy Monitoring strategy is a crucial component of our framework, because it enables the formulation and validation of other strategies. By verifying the energy efficiency improvements, through profiling tools, strategies can be applied iteratively and consequently adapted. Energy Monitoring also allows to take into account other parameters, e.g. the software mission and its main functionalities, the required quality of service, and the interests of the stakeholders. For example, reducing the network usage might improve energy efficiency, but it might also violate service level agreements on response time or availability.

[2]`http://www.google.com/green/bigpicture/`, last visited on November 12th, 2014

[3]`http://www.microsoft.com/environment/IT_Energy/IT_Energy.aspx`, last visited on November 12th, 2014

6.5.2 Refactoring: identify and remove energy inefficiencies

Predictive models embed the knowledge about both the resources (e.g., CPU) that consume power and the activities (e.g., disk transfers) that drive their consumption. The aim of the Refactoring strategy is to identify those code patterns responsible for high energy usage. Taking inspiration from the well-known book of Fowler and Beck [37] we call these code patterns *Energy Code Smells* [131], i.e. implementation choices (at code, design or architectural level) that make the software execution less energy efficient.

The Refactoring strategy is backed up by empirical evidence. In Chapter 3 we already presented two best practices for developing energy-efficient software, extracted from our wiki[4], and we proved their impact in terms of energy efficiency improvements. Moreover, in a previous study on an embedded system [131], we found occurrences of five distinct Energy Code Smells, selected among those detected by a well-known automatic static analysis tool (CppCheck [5]). The refactoring of such smells successfully improved the energy efficiency of the tested code. However, as software execution depends not only on its internal structure and host environment but also on the input it receives, the Refactoring strategy may show its results only in specific situations. For this reason, before applying the Refactoring strategy, the most frequent usage scenarios have to be identified. These scenarios will provide the most promising candidates for refactoring activities.

In the remainder of this section we provide some examples of guidelines for the Refactoring strategy. These guidelines were obtained by combining our experience together with the evidence provided by similar works ([42], [94], [122] and [26]).

Clean up useless code and data.
As software evolves, many parts may become obsolete. Writing to never-read variables and other useless routines (e.g., repeated conditionals) may consume power purposelessly. Cleaning up these instructions might improve energy efficiency, as well as maintainability. Many static analysis tools are able to detect useless code.

Look for Immortals.
The lifecycle of software processes and threads must be carefully managed. The Immortality Energy Smell describes situations where a software service restarts after explicitly being killed by the user, continuing to drain energy. Sometimes, software immortals are created on purpose: in this cases, death and rebirth phases

[4] `https://wiki.cs.vu.nl/green_software/index.php/Main_Page`
[5] `http://cppcheck.sourceforge.net/`

of the processes/threads should be as graceful as possible, in order to reduce the resource usage overhead and the consequent energy waste.

Focus on higher-level structures and complex routines
Like in performance optimization problems, improvements obtained at lower level might be hidden from higher level inefficiencies. This is especially true when there are many software layers or when software runs in a complex environment (e.g., virtualization, distributed systems). Start refactoring from higher level constructs: their impact on CPU and memory (and consequently, energy) is significantly higher compared to basic data types.

Do not trust loops.
Loop constructs are powerful, but their contents must be carefully monitored. Loop smells happen when an application repeats the same activity on a loop, without achieving the intended results and uselessly consuming energy (e.g., polling an unreachable server). Detecting and refactoring such loops can save a significant amount of energy, especially on battery powered devices.

Reduce amount of data transferred.
In distributed and high-performance systems, or in battery powered devices using power-consuming radio transmission, data transfer might be a significant source of power drain. Data exchanged between software applications and/or databases (local or remote) can be optimized using data compression or data aggregation techniques. The energy impact of this optimization might be crucial, in data-intensive and Big Data applications. An example of an useful metric to monitor is Communication Energy Cost [114], that estimates the energy consumption induced by data transfers for each software component.

6.5.3 Self-adaptation: energy efficiency by design

While the Refactoring strategy is useful to developers who aim at increasing the energy efficiency of their existing applications, the Self-Adaptation strategy is more suitable when building software from scratch. The key idea is to provide different configurations of the same application, to be selected according to the best trade-off between provided features and consumed energy.

Actions needed to implement self-adaptation depend on the usage context of the application. Hence, we need to profile the application energy usage in different usage scenarios, i.e. identifying “Energy Profiles” which will provide either the full set or a subset of functionalities. This approach is compatible with the Refactoring strategy, as seen in Section 6.5.2, and also requires the

	Profile 1			Profile 2	
Software Sensor	State (ON/OFF)	Refresh After (s)	Software Sensor	State (ON/OFF)	Refresh After (s)
PhoneSensor	ON	3600	PhoneSensor	ON	600
LocationSensor	OFF	-	LocationSensor	ON + GPS ON	600
WiFiSensor	OFF	-	WiFiSensor	ON	600
BluetoothSensor	OFF	-	BluetoothSensor	ON	600
DeviceInfoSensor	ON	3600	DeviceInfoSensor	ON	3600
DeviceStatusSensor	ON	3600	DeviceStatusSensor	ON	600
DeviceSettings	ON	3600	DeviceSettings	ON	600
TerminalActivity	OFF	-	TerminalActivity	ON	600
DataSensor	ON	3600	DataSensor	ON	3600

Table 6.1: Example of Configuration file for Self-Adapting Applications

Energy Monitoring strategy to properly classify the energy usage of the different scenarios (see Section 6.5.1). In [3] is provided an example of a self-adapting mobile application. The application reconfigures itself based on the remaining battery level. Authors implemented self-adaptation working on the application functionalities, enabling or disabling modules and tuning parameters such as the time granularity of the data collected from the device and transferred to the server. Table 6.1 shows an example of different profiles for the mobile application.

Results show improvements up to 30% compared to an equivalent, non-adaptive application. Improvements depend on the scenario and on the level of trade-offs that developers are willing to reach.

Compared to the Refactoring strategy, the Self-Adaptation strategy introduces a relevant set of changes to the software system. While Refactoring mostly operates at code level, Self-Adaptation has also a relevant architectural impact. Raibulet et al. [101] proposed a set of architectural metrics to evaluate the adaptivity of a software system. Although those metrics are not specific for energy-driven Self-Adaptation, they can be adopted as a reference for developers that want to introduce self-adaptive mechanisms in their application. For example, the *MaAC (Minimum architectural Adaptive Cost)* expresses the fixed cost of adaptivity at architecture level.

6.6 Conclusions

Energy efficient software is a challenging topic that involves complex trade-offs among stakeholders. From a technical perspective, several tools and best practices are available, although they are not yet well integrated in an organic framework able to provide the software developers and designers a unifying view.

In this chapter, we addressed this problem by providing a conceptual framework that provides an high-level view over the possible operational strategies for energy-efficient software. We described three strategies, i.e. Energy Monitoring, Refactoring and Self Adaptation, although other strategies, at technological or process level, can be plugged in the framework, provided that they have a measurable impact on energy consumption.

This contribution answers RQ 4, namely "Can we provide strategies to improve software energy efficiency?" We proved that the current State-of-the-Art, backed up by empirical evidence, is mature enough to provide such strategies. In the next chapter, we will give an example of a possible improvement of the Refactoring strategy: specifically, we will provide an approach to identify more general issues in software energy efficiency (*energy hotspots*), not just at code level, but at multiple levels of abstraction.

7 The GREENSWEEP Approach for Software Energy Efficiency Research

Software energy efficiency is a pioneering research topic where empirical experimentation is widely adopted. Nevertheless, current studies and research approaches struggle to find generalizable findings that can be used to build a consolidated body of knowledge for "green" software. In this chapter, we identify the issues that characterize the research in software energy efficiency. Then, we propose an approach to systematically identify software energy efficiency issues (hotspots) in software applications. We called this approach GREENSWEEP (Guided REcognition and EvaluatioN of SoftWare EnErgy hotsPots). The GREENSWEEP approach combines traditional hypothesis-driven/top-down research with a bottom-up discovery process using data mining techniques. We also discuss the implications of GREENSWEEP on the traditional characteristics of empirical experimentation. In the long run, we foresee that the experimental findings discovered through our approach will be more generalizable and reusable to design and develop energy-efficient software. This chapter is related to RQ 4.

7.1 Introduction

Current research in software energy efficiency lacks of well-defined, validated methods: although there is a significant amount of scientific works in the field (see Section 3.2 in Chapter 3), to date they show limitations and lack of generalizable principles and results [71]. The research community in software energy efficiency already highlighted that the field is characterized by peculiar issues. Among them:

1. *High complexity.* In order to improve energy efficiency, the relationship between software operations and the energy consumption of the underly-

ing hardware has to be clearly defined. Hence, the research is inherently multi-disciplinary [18], characterized by a very large amount of variables to control, especially due to the high heterogeneity of IT devices and platforms (e.g. mobile devices, cloud-based architectures, embedded systems). For this reason, it is very difficult to trace software behavior *directly* to the used hardware [110].
2. *Anecdotal evidence.* Many empirical studies have been conducted to assess the factors that determine software energy efficiency. However, current evidence is mostly anecdotal and insufficient to provide generalized principles. This becomes clear from the contradictions in the conclusions drawn by researchers in different empirical studies (e.g. Cameron [17] concludes that energy efficiency and performance positively correlate, while Capra [19] concludes the exact opposite).
3. *Lack of a unified approach.* Different research communities have tried to tackle the problem of energy efficiency within their own expertise, at the cost of precision in other domains. Hence, the studies lack representativity, and are characterized by heterogeneous techniques and analysis methods [71]. We argue that this prevented from building sound evidence. An historical parallel is with object-oriented software design in the late 80's, when a plethora of different notations were proposed, making it impossible to compare designs expressed in different languages, test them or validate their consistency. This problem was addressed when finally the OMG standardized UML.

Above peculiarities of energy efficient software research delineate requirements for an ad hoc approach for the topic. In this chapter, we propose a mixed approach for experimentation in energy-efficient software research with the aim to address those threats and ultimately speed up the identification of software-related properties that impact energy consumption.

This chapter is organized as follows: in Section 7.2, we describe our proposal for experimentation in software energy efficiency research. In Section 7.3, we discuss its implications for empirical experimentation. Section 7.4 concludes the chapter.

7.2 The GREENSWEEP Approach

From experience, we developed a mixed approach to conduct experimentation for energy-efficient software. The approach is called GREENSWEEP (Guided REcognition and EvaluatioN of SoftWare EnErgy hotsPots). Currently, we are applying GREENSWEEP in the context of national research projects and we plan to develop it further on a larger scale. In Fig. 7.1, we give a graphical

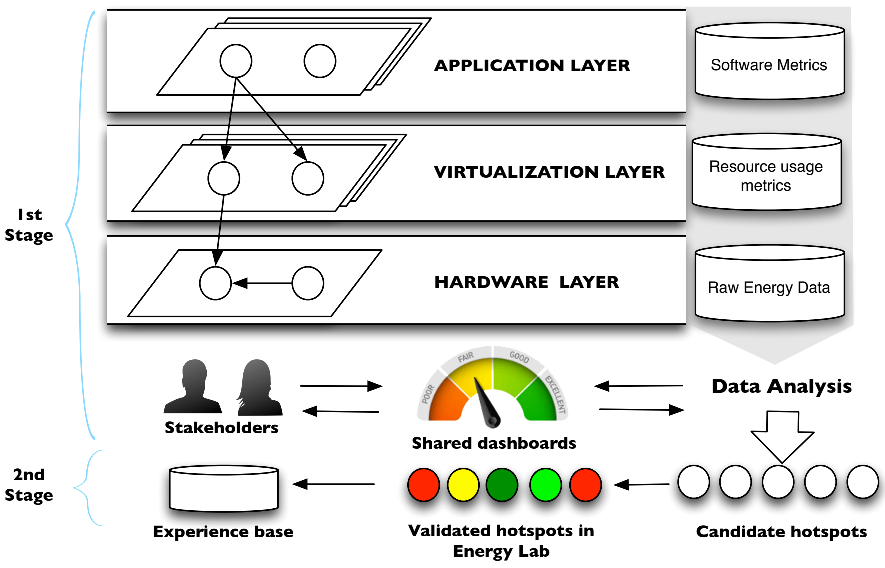


Figure 7.1: Overview of the GREENSWEEP approach.

overview.

7.2.1 Background: Energy Hotspots

The main operational goal of GREENSWEEP is to identify *energy hotspots* (as circles in Fig. 7.1). The concept of *hotspot* has already been introduced from a software architecture perspective in a crosscutting sense, both in performance [135] and evolution [100], as actionable points of interest, crucial for a certain property. Recently, researchers in software energy efficiency have introduced the term *Energy Bug*, defined as "an error in the system, either application, OS, hardware, firmware or external, that causes an unexpected amount of high energy consumption by the system as a whole" [94]. However, this very broad definition has a questionable 'negative' connotation: high energy consumption is not necessarily due to an error, rather to trade-offs with other qualities (e.g. higher performance). The alternative concept of *Energy Smell*, defined in the previous chapter as "an implementation choice that makes the software execution less energy efficient" [131], has, instead, a 'positive' connotation. In this case, the focus is on software: higher energy consumption is the result of an (explicit or implicit) implementation choice, hence not necessarily an error. This definition, however, focuses only on the implementation level, while anomalies in energy consumption can be identified at any level of abstraction, as from the previous definition. Building up from this, we define *energy hotspots* as elements or properties, at any level of abstraction of the architecture, that have a measurable and significant impact on energy consumption. In particular, our research focuses on *software* energy hotspots, i.e. software-intrinsic properties. For this purpose, we consider hardware as part of the context that we precisely model to better define the scope of our findings (see Section 7.3).

7.2.2 1st stage: Hotspot Identification

To identify *software* energy hotspots, we apply different development practices, design techniques, architectural tactics/patterns, etc. on software applications. Considering a software application as a set of independent variables, a particular software configuration can be defined as a treatment [137], i.e. a set of values assigned to the corresponding independent variables. We plan to instantiate several versions of the same software application on multiple virtual machines (VMs) running on several servers. Each version of the application will differ in terms of software configuration. Running the applications in VMs enables us to enhance the scale of the experimentation. In this stage, we monitor the following three layers (see Fig. 7.1, top):

1. In the application layer, we monitor software events and use instrumented code to retrieve software measures that can be relevant for energy consumption (e.g. response time, served requests, data transfer rate).
2. In the virtualization layer, we monitor VM allocation and migration events, plus other relevant measures from the OSs running in the VMs (resource usage data, e.g. CPU, RAM, or system load measures like running processes and threads).
3. In the hardware layer, we monitor power consumption data from the physical servers, depending on the available measurement infrastructure: values could be per-rack, aggregated per machine, or even broken down to every single component.

Monitoring software behavior in the first two layers and associating the results with the energy measures from the hardware layer allows to identify, locate, and characterize the software energy hotspots. Energy hotspots may be located on software architecture (e.g. architectural elements, structures, patterns), source-code (e.g. libraries, classes, methods), OSs (RPC, system-calls, services) or resource usage (RAM, CPU, I/O devices, data management/storage, network). This results in a large number and type of measurements and, thus, understanding which data to collect first is not an easy task. For this reason, we started collecting data sources connected to identified green practices (but not fully evaluated yet) in the EFRO MRA Project Cluster Green Software[1] and our library of Green ICT practices[2] in the different layers.

While these layers are a conceptual representation that helps us in modeling the environment under observation, indeed many abstraction and infrastructure layers exist in modern software systems. With such a complexity, we believe that discovering energy hotspots with the traditional experimentation approach (i.e. setting up real controlled environments and assign treatments to different configurations) is unfeasible, due to the high number of factors, scattered among the layers.

There is a parallel in social sciences where real-world phenomena are difficult to be evaluated statistically: there are so many variables that no regularities can be found by simply looking at the variables. In that case, one alternative to statistical research is case study research, where social scientists study phenomena in one case (e.g. energy usage of one piece of software) and trace the mechanisms [133] by which these phenomena were produced. However, even the replication of a new case will have a slightly different structure and may contain other mechanisms: in that case, it is possible to generalize about the mechanisms, not the cases. It can be analyzed, explained, and predicted what the effect of a single mechanism is, but how all mechanisms interact in a particular case is not predictable in general. However, in software energy efficiency research, although the interaction of the many involved variables is likewise complex and not yet fully understood (see Section 7.1), those interactions are produced by mechanisms which are based on physical properties of the hardware components and so they should be deterministic.

For this reason we propose the use of data mining and analysis techniques, on the data gathered from each layer, to discover patterns for *candidate* software energy hotspots. For example, suitable data mining techniques for this task include decision trees for exploration, neural networks for outliers prediction, and subsequently k-neighbors and clustering algorithms in n dimensions to find recurring patterns. Data analysis results are continuously validated with the stakeholders who own the domain knowledge (e.g., data centres administrators, developers). To speed up the feedback cycle, we propose to use shared dashboards that contain interactive graphs, checkboxes and fields to input further comments: this information can be analyzed by researchers and (partly) automatically fed back to the data mining tools for better tuning (e.g. in neural networks, increasing the weight of certain branches). In practice, such mechanism enables an iterative and supervised knowledge acquisition process, which is faster than the traditional experimentation, where usually feedback is downstream after experimentation result.

[1]http://www.clustergreensoftware.nl
[2]http://greenpractice.few.vu.nl

7.2.3 2nd stage: Hotspot Verification

After the identification, location and characterization of the candidate hotspots and the mechanisms that possibly produce them, we reproduce these mechanisms and their effects in isolation in the lab. This is done in a dedicated Energy Lab, where the second part of the research takes place (see Fig. 7.1, bottom part). The Energy Lab is a shared laboratory among our partners where to perform experiments to correctly assess the significance and impact of the hotspots. A prototype of Energy Lab is the Software Energy Footprint Lab (SEFLab [34]) where we are currently conducting our experiments in software energy efficiency as part of a national research project. The second stage is necessary to recover the experimentation rigor relaxed in the first stage (see Section 7.3 for further discussion). It also permits to study in a more controlled environment the identified hotspots (in Fig. 7.1 the different color scales represent the levels of impact on power consumption) and the causal relationships involved.

The **validated** hotspots in Fig. 7.1 are those candidate hotspots confirmed as valid (i.e. whose effect is proven in a predictable way for well-defined context variables) at the end of this in-depth validation. They contribute to the expansion of the current Experience Base.

7.3 Research Implications

The peculiarities of software energy efficiency research as well as our approach to conduct such research have a number of implications on the evidence-based principles. Based on the current state of the art, we elicited a list of ten properties that, to a large extent, an experiment in SE should aim at. Table 7.1 illustrates which of these properties can be either relaxed or stressed according to the discussed research methodology for software energy efficiency research.

Table 7.1: Experimental Software Engineering Properties

Properties to relax	**Properties to stress**
Hypothesis-driven	Contextualization
Statistical significance	Cause-and-effect analysis
Controlled	Randomized assignments
Blocked subjects assignment	Replicability
Balanced subject groups	Competing alternatives

The approach we presented is based on exploratory data analyses as a preparation for (more traditional) in-depth experimentations (see also Section 7.2). Accordingly, we relax rigor in favor of pragmatism during the first stage of our evidence-based research endeavor. This translates in a bottom-up attitude, which requires avoiding to build formal hypotheses to test, postponing accuracy further in the process when rigor is recovered in the Energy Lab. As a consequence, in the first stage, most requirements for rigorous experimentation have to be relaxed as well. We assign, for example, treatments to a specific context and observe the environments through their measurable properties, despite we still do not have full control over all the input variables due to complexity. In a similar way, although we are able to perform randomization of subjects through virtualization, we still need to relax the requirement to perform blocked assignments or balancing subject groups. That is, the data collection and analysis procedure is, or should be, opportunistic to focus on an inherently complex universe of variables. This produces large observations derived from real situations rather than cleaner but smaller data from a rigid framework of experimentation, which is often not a choice but a real constraint when dealing with cloud infrastructures. We have similar constraints even in mobile devices where the nested virtualization makes it impossible to separate the software layers for exact experimentation. This variability requires modeling the operational environment under observation, an operation that we call *contextualization*. Defining the right context variables and their dependencies is one of the most challenging but important task we are facing (as also stressed by a recent roadmap for Empirical Software Engineering [10]). A detailed enough model for the context (which requires modeling both hardware and software at different levels of abstraction) permits to accurately establish the degree of similarity between two contexts (e.g. two mobile devices, two system architectures or implementation choices) in the comparison of power consumption measurements. Also, replicability of analysis is improved in terms of context reproduction.

Of course, high contextualization not only increases the awareness of the confounding factors, but it is also a prerequisite to control the internal validity of the experiments and, thus, the accuracy of the measurements. The fact that the measurements cannot be only hypothesis-driven, however, hampers the possibility to control the construct- and the external validity, and this eventually lowers the conclusion validity. We therefore need to relax those types of validity in trade for a more pragmatic approach in the short run: this allows to populate a larger results set, that serves to trigger further investigations following the traditional experimental SE research approach in the Energy Lab. In the long run, however, the rigorous contextualization will define clear boundaries and impacts of the hotspots. In turn, we expect that the construct and the external validity will increase while the internal validity will be better controlled in the short run.

In summary, in relation to the peculiarities listed in Section 7.1:

1. We reduce the *complexity* of the software/hardware interaction via abstraction, i.e. by capturing the most important concepts and relationships via a precise contextualization of the environment. This allows to identify can-

didate hotspots, which will be verified later in the Energy Lab through traditional experimentation.

2. We mitigate the problem of *anecdotal evidence* with in-depth verification steps in the Energy Lab on a reduced list of candidate energy hotspots.
3. In relation to the *lack of the unified approach*, thanks to randomization in multiple subjects, contexts and hardware configurations at the same time, our approach allows in its first stage to gather candidate measures for varieties of scenarios; at the same time, by configurable contextualization we can potentially cover different approaches typically limited to a single level of abstraction or execution environment.

7.4 Conclusions

Empirical experimentation for software energy efficiency is still an emerging topic. In this chapter, we have identified three main related issues: high complexity of the topic which involves multi-disciplinary competences, lack of strong evidence to contribute generalized principles to the body of knowledge, and the lack of a unified research approach. These three issues point to a more general problem where the traditional way of conducting experimental software engineering is not suitable anymore for software energy efficiency research.

To tackle this problem, we presented in this chapter a mixed approach: first we discover hypotheses and patterns by using data mining and analysis techniques, continuously tuned with fast feedback cycles with stakeholders; in a second stage, we can follow again the traditional experimental approach intended to test the hypotheses. We have discussed some implications of our approach on a set of properties for empirical experimentation.

This chapter is related to RQ 4, namely: “Can we provide strategies to improve software energy efficiency?” The GREENSWEEP approach we presented can be seen as a possible improvement of the Refactoring strategy presented in Chapter 6. However, GREENSWEEP has not yet received proper validation, hence we cannot claim that it answers our RQ. For this purpose, we are applying it in ongoing research efforts and future work will be devoted to further develop GREENSWEEP as an open contribution to the research community.

8 Conclusions

Software has a relevant impact on the energy consumption of ICT devices. Energy-efficient software is crucial for extending the lifetime of battery-powered devices and for reducing the ICT environmental impact. However, software engineering does not provide a consolidated body of knowledge neither on how to improve the energy efficiency of existing software systems, nor on how to design and develop energy-efficient software. This thesis tries to fill this gap through empirical experimentation. Throughout its chapters, we traversed the abstraction levels that divide software from hardware and we provided tools, guidelines and approaches for analyzing and improving the energy efficiency of software applications. In this chapter, we summarize our main contributions, with respect to the research questions presented in Chapter 1. We conclude this dissertation with our future research agenda.

8.1 Main Contributions

The goal of this thesis is to provide a body of knowledge that supports the engineering of energy-efficient software systems and applications. Hence, the main Research Question for this thesis is *"How can we engineer energy-efficient software?"* In Chapter 1, we identified four sub-Research Questions that further characterize our research problem. This section summarizes the answers to these questions, according to our findings.

8.1.1 RQ 1. What is the correlation between software and hardware energy consumption?

In order to define the relationship between software and hardware energy consumption, we first need to determine if hardware resource usage is relevant for energy usage. This is the subject of RQ 1a, namely *"RQ 1a. Is hardware re-*

source usage correlated with energy consumption during software execution?". In line with our empirical approach, in Chapter 2 we present an experiment conducted on Desktop computer systems. The results show that CPU and memory usage have the highest correlation values with energy consumption, but those vary significantly depending on the usage scenario. We also compared the impact of software usage on two machines of different generations. Our results show that modern hardware resources are more energy-efficient in idle states, but more consuming in intensive operation modes. This makes the role of software even more important, in order to use resources efficiently and reduce energy waste. Hence, our initial claim in Chapter 1 is verified by means of empirical evidence.

To make the relationship between hardware and software more transparent, we need to make this knowledge actionable. That is, we have to verify whether resource usage information can be used to provide estimations and prediction on energy consumption and relate it to software behavior to steer its development and engineering. That is the focus of RQ 1b, namely *"How can software properties be used as a predictor for hardware energy consumption?"* We surveyed the academic literature and came up with a classification of approaches to measure and model software energy consumption. Indeed, using resource usage as a predictor is a viable option: many tools and techniques are already available and ready for use on different platforms (e.g. mobile, embedded systems, laptops). Through these tools, developers and users are able to gain insights on the energy impact of the software they use or produce, with a reasonable level of accuracy. However, what is still missing is the integration of these models into software applications, to realize proper energy-aware behaviors.

8.1.2 RQ 2. What is the impact of using best practices for software energy efficiency?

The first level of abstraction we consider for our research on software energy efficiency is the source code. The aim of RQ 2 is to assess to what extent coding with energy efficiency in mind can make a difference on the actual energy consumption of an IT device. For this purpose, we performed an empirical experiment where we applied two best practices for energy-efficient software development, extracted from various sources in academic literature and industrial practice, on two widely used open source software applications. Our results show that applying the practices allows to save up to 25% energy consumption, and it increases the energy proportionality of software behavior as well.

This stresses the importance of supporting developers in implementing energy-efficient software. By formally describing these software practices in the form of code patterns, it would be possible to automatically detect and refactor energy inefficiencies during development, similarly to what is already done for other quality aspects (e.g. performance, security, reliability).

8.1.3 RQ 3. How can software architectural solutions realize energy efficiency?

After assessing the potential impact of energy efficient software development, our next step was to increase the level of abstraction and address energy efficiency at the architectural level, as with other more traditional software quality aspects. We first systematically surveyed the state-of-the-Art to answer RQ 3a, namely: *"Are there software architectural solutions that address energy efficiency aspects?"* We designed and conducted an SLR on architectural solutions for energy efficiency in Cloud-based software applications. From our results we were able to identify a number of solutions, that we generalized in architectural strategies, techniques and components for energy efficiency. Then, we addressed RQ 3b, namely *"How can architectural solutions for energy efficiency be made reusable?"* extracting from our results architectural *tactics* for energy efficiency. Those tactics not only support software architects into creating energy efficient software architectures, but also show that energy efficiency can be indeed realized at architectural level and can thus be considered a software quality aspect.

8.1.4 RQ 4. Can we provide strategies to improve software energy efficiency?

The previous RQs analyzed the problem of software energy efficiency at different levels. This effort needs to be complemented with a suitable conceptual framework that models the different outcomes and connects them towards our goal. That is the aim of RQ 4. In Chapter 6, we reflect upon our empirical evidence and present such a framework, together with examples of high-level strategies for engineering energy efficient software. In Chapter 7, we present a preliminary example of an empirical approach (GREENSWEEP), based on a strategy extracted from our framework, to identify energy efficiency issues in existing software applications.

8.1.5 Answering the Main Research Question: lessons learned

The main Research Question that drives this thesis is *"How can we engineer energy-efficient software?"*

Indeed, we selected an ambitious and challenging problem, in an emerging but pioneering research field. Nevertheless, by using *divide et impera* and reasoning by increasing abstractions, we learned valuable lessons that can help in solving our main research question (see Chapter 1). Keeping in mind the following lessons is necessary to engineer energy-efficient software.

1. *Energy consumption is software-defined.* Although hardware technologies continuously improve in energy efficiency, ICT energy consumption is still rising: idle consumption has decreased, but in high-performance modes new-generation devices consume more than previous ones, as proven by our empirical evidence. Moreover, we are moving more and more towards a model where hardware is a commodity that can be provisioned on demand through software. Examples are software-defined networks and datacenters. This important transition has a fundamental consequence: the role of software will be more and more relevant in driving energy consumption. Hence, models based on resource usage to predict energy consumption will be extremely valuable in determining waste of resources such as CPU and memory, the primary reason for energy inefficiency.

2. *There is no one-size-fits-all.* When analyzing software energy efficiency, the importance of usage scenarios is crucial. It is very likely that the energy efficiency of a software application varies according to the specific task at hand. Also, it is imperative to carefully model the context: due to the complexity of software and hardware interaction, there are many factors playing a role in energy consumption. For this reason, for example, some solutions that improve energy efficiency in mobile devices might not be effective in other systems.

3. *No improvement is possible without measurement.* Energy consumption caused by software can be odd and counter-intuitive. For this reason, empirical validation is extremely important in this field. In this thesis, we showed examples of second-order effects of coding practices that can *neglect* improvements in energy efficiency. To avoid these second-order effects, guidelines and best practices need to be carefully validated.

8.2 Future work

This section concludes this thesis, but our research is far from completed. Software energy efficiency has just started gaining momentum in the scientific community and many challenges lie ahead in the form of new research questions.

As a stable part of our research agenda, we will keep on performing empirical experimentation to validate the best practices for energy-efficient software. This activity will contribute to establish a solid knowledge base for developers and architects. In the meantime, we are already working on representing this knowledge as an ontology of concepts and entities, using semantic technologies. This has potential for multiple applications: firstly, this abstract modeling will allow us to automate the process of applying practices to existing applications. Moreover, we can explore the impact of energy efficiency on other software quality attributes [72]. This trade-off analysis is needed for the inclusion of energy efficiency in a comprehensive software quality model.

Finally, our main commitment in the next years will be devoted to provide education in software energy efficiency. During the last three years, we were able to bring awareness on the subject and we participated in successful projects (such as the MRA Cluster Green Software) that created a network of stakeholders, both companies and public institutions, interested in software energy efficiency. Together with these stakeholders, we assessed the need of professionals in the European market with competences and skills in Green IT and Green Software Engineering. This need was the main motivation for the new master track in “Software Engineering and Green IT” of the VU University Amsterdam, which aims to educate professionals in Software Engineering with specific skills and knowledge in IT sustainability issues [70]. In the context of this master track, we will provide a course focused on experimentation in software energy efficiency, called Green Lab. In this course, students will follow the path traced by this thesis, performing experiments to validate practices and case studies provided by the industrial stakeholders.

Driving the ICT industry towards a more sustainable path requires a solid, long-term effort in both research and education, involving national and international institutions. This thesis is a step further in such a direction.

English Summary

The energy consumption of ICT is growing at an unprecedented pace. The main drivers for this growth are the widespread diffusion of mobile devices and the proliferation of datacenters, the most power-hungry IT facilities. In addition, it is predicted that the demand for ICT technologies and services will increase in the coming years. Finding solutions to decrease ICT energy footprint is and will be a top priority for researchers and professionals in the field.

As a matter of fact, hardware technology has substantially improved throughout the years: modern ICT devices are definitely more energy efficient than their predecessors, in terms of performance per watt. However, as recent studies show, these improvements are not effectively reducing the growth rate of ICT energy consumption. This suggests that these devices are not used in an energy-efficient way. Hence, we have to look at software.

Modern software applications are not designed and implemented with energy efficiency in mind. As hardware became more and more powerful (and cheaper), software developers were not concerned anymore with optimizing resource usage. Rather, they focused on providing additional features, adding layers of abstraction and complexity to their products. This ultimately resulted in bloated, slow software applications that waste hardware resources – and consequently, energy.

In this dissertation, the relationship between software behavior and hardware energy consumption is explored in detail. For this purpose, the abstraction levels of software are traversed upwards, from source code to architectural components. Empirical research methods and evidence-based software engineering approaches serve as a basis. First of all, this dissertation shows the relevance of software over energy consumption. Secondly, it gives examples of best practices and tactics that can be adopted to improve software energy efficiency, or design energy-efficient software from scratch. Finally, this knowledge is synthesized in a conceptual framework that gives the reader an overview of possible strategies for software energy efficiency, along with examples and suggestions for future research.

Nederlandse samenvatting

Het energieverbruik van ICT groeit met een ongekende snelheid. De belangrijkste redenen voor deze groei zijn het wijdverspreide gebruik van mobiele apparatuur en de toename van het aantal datacenters, de meest energie behoeftige IT-faciliteiten. Daarnaast wordt voorspeld dat de behoefte aan ICT technologien en services in de aankomende jaren zal toenemen. Het vinden van methoden om de ICT energie footprint te verkleinen is en zal topprioriteit zijn voor onderzoekers en professionals.

Hardware technologie is de afgelopen jaren substantieel verbeterd, gemeten in prestatie per watt is de huidige ICT apparatuur meer energie efficint dan zijn voorgangers. Echter, recent onderzoek toont aan dat deze verbeteringen de groei van het ICT energieverbruik niet effectief verminderen. Dit wijst erop dat de apparaten niet op een energie-efficinte manier worden gebruikt. Daarom moeten we naar de software kijken.

Moderne software applicaties worden niet ontworpen en gemplementeerd vanuit een energie efficint perspectief. Terwijl hardware steeds krachtiger (en goedkoper) werd, waren de software ontwerpers niet bezig met het optimaliseren van het bron verbruik. In plaats daarvan focusten zij op het toevoegen van extra eigenschappen, wat leidde tot extra lagen van abstractie en complexiteit in hun producten. Uiteindelijk leidde dit tot opgeblazen, langzame software applicaties welke hardware capaciteiten, en dus energie, verspilden.

Uiteindelijk leidde dit tot opgeblazen, langzame software applicaties welke hardware capaciteiten, en dus energie, verspilden. Hiervoor werden de abstractielevels van software overspannen van broncode naar structurele componenten. Empirische onderzoeksmethoden en evidence based software engineering zullen hiervoor als basis dienen. Allereerst toont dit proefschrift de relevantie van software voor het energieverbruik van hardware. Daarnaast geeft het voorbeelden voor best practices en tactieken die te gebruiken zijn om software energie efficintie te verbeteren of te ontwerpen. Tot slot wordt deze kennis toegepast in een conceptueel kader welke de lezer een overzicht geeft van de mogelijk strategien voor energie efficinte software, met daarbij voorbeelden en suggesties voor toekomstig onderzoek.

Acknowledgements

As I glance at my doctoral endeavour, I cannot help but notice a common theme: duality. I always thought of myself as having two different “souls”: a methodical, engineering one, and a more creative, abstraction–oriented one. This somehow reflects also in the choices I made and in the path that led me here.

Obviously, this duality is primarily represented by my two *almae matres*: Politecnico di Torino and VU University Amsterdam. Consequently, I want to begin by thanking my two supervisors: Maurizio and Patricia. They were both great supervisors, but in different ways: Maurizio inspired me with his approach to software engineering as “the equivalent of an experimental physicist”, teaching me experimentation and the rigour of quantitative evidence. Patricia taught me how to challenge my beliefs, or rather that “we don’t believe”, showing me how a true scientist must behave: be bold when stating facts but be ready to disprove them. I consider both of them not only amazing researchers, but great people. I am glad to have had the opportunity of working with them during my PhD.

I would also like to thank my thesis committee, for their precious feedback and support.

I would definitely not be able to write these words without the help and support I received from my colleagues, in Italy, the Netherlands, and everywhere else. For this reason, I want to thank the team of the Lab “Italo Gorini” of the Politecnico di Torino, Luca Ardito, Federico Tomassetti, Andrea Martina, Syed Ali, Najeeb Ullah, Oscar Rodriguez, Christian Figueroa, and all my PhD colleagues of the XXVII cycle. A special mention goes to Antonio Vetrò, my master thesis supervisor, long-term collaborator and German beer provider. If it wasn’t for him, I would never have thought about software energy efficiency. I still don’t know whether I should thank him for that.

A good thing about working in two universities is that you work with twice as much people, and just as amazing. My colleagues at the VU University Amsterdam, for example: my former colleagues Maryam Razavian, Damian Andrew Tamburri, Christina Manteli, Hector Fernandez, as well as Han van der Aa, Fahimeh Alizadeh, Nelly Condori-Fernandez, Karl Lundfall, Klaas Andries de Graaf and everyone else I forgot to mention.

The administrative staff of both universities deserve a special mention for their help and support. In particular, I want to thank Mojca, Caroline and Elly from the VU for helping me solve the many problems a foreign student has to face.

My duality comes back again in the choice of my paranymphs. Marco Nicotra, computer engineer, and Matteo Raimondi, phylosopher and writer, are two of my best friends, and also represent for me two completely different ways of living life. I was so lucky to learn from both of them in my life, several times. Thanks guys. With them, I want to thank my friends from Palermo, who always supported me –

especially by contradicting me. Pietro, Dario, Manfredi, Roberto, Sergio, Sergio, Francesco: grazie. You taught me what friendship really means, and I will never forget that. I also want to mention my friends, flatmates and fellow students at the Politecnico: Vincenzo and Gaetano. Thanks guys, for sharing our great time in Turin (and for bearing with me in Largo Orbassano). Here in Amsterdam, I was again very lucky: I made friends who never made me feel homesick. Paolo, Giacomo, Jennifer, Justine, all of Bozzellions, and my band ViciousK: thanks guys, I'm looking forward to continue the adventure of living in Amsterdam with you. For brevity reasons, I obviously cannot mention everyone: all the friends I made in my journey, in Palermo, Torino, Cordoba, Amsterdam, would deserve to be mentioned for sure.

However, a special mention for a special person: Joana, thanks for your love and caring support. We make a great team!

Con queste ultime parole in Italiano, voglio ringraziare la mia famiglia, a Palermo, Napoli, Milano, e in giro per l'Italia. In particolare voglio menzionare le due persone che sono state sempre presenti nella mia vita, anche quando lontane, e che mi hanno sempre dato il loro affetto e sostegno incondizionato. Anche con loro, il tema della dualità ritorna: mio padre, Placido, mi ha trasmesso nelle vene la curiosità e la passione per l'ingegneria sin da bambino. Vederlo orgoglioso di me mi rende felice. Mia sorella, Elsa, ha sempre cercato di farmi pensare fuori dagli schemi. Lei é un esempio per me ed in fondo, ha sempre avuto ragione. Come quando mi ha convinto a fare il dottorato. Sono sicuro che con mio nipote, Antonio, farà un lavoro ancora migliore di quanto non abbia fatto con me.

Infine, voglio ringraziare un'ultima persona. Anche lei c'é sempre stata, anche se non c'é più.

A mia madre
Giuseppe Procaccianti
March 6th, 2015

Bibliography

[1] Acpi 5.1 specifications, July 2014. Available from: `http://www.uefi.org/sites/default/files/resources/ACPI_5_1release.pdf`. (Cited on pages 17, 18, and 19.)

[2] Luca Ardito, Giuseppe Procaccianti, Marco Torchiano, and Giuseppe Migliore. Profiling power consumption on mobile devices. In *ENERGY 2013, The Third International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, pages 101–106, 2013. (Cited on pages 104 and 105.)

[3] Luca Ardito, Marco Torchiano, Marco Marengo, and Paolo Falcarin. gLCB: an energy aware context broker. *Sustainable Computing: Informatics and Systems*, 3(1):18–26, 2013. `doi:10.1016/j.suscom.2012.10.005`. (Cited on page 111.)

[4] AT&T. Application resource optimizer (ARO), 2011. Last visited: May 5th, 2014. Available from: `https://developer.att.com/application-resource-optimizer`. (Cited on page 108.)

[5] Woongki Baek and Trishul M Chilimbi. Green: A framework for supporting energy-conscious programming using controlled approximation. In *Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '10, pages 198–209, New York, NY, USA, 2010. ACM. (Cited on page 44.)

[6] Rami Bahsoon. A framework for dynamic self-optimization of power and dependability requirements in green cloud architectures. In *Proceedings of the 4^{th} European conference on Software architecture*, ECSA'10, pages 510–514, Berlin, Heidelberg, 2010. Springer-Verlag. (Cited on page 86.)

[7] Nisa Bakkalbasi, Kathleen Bauer, Janis Glover, and Lei Wang. Three options for citation tracking: Google Scholar, Scopus and Web of Science. *Biomedical digital libraries*, 3(1):7, 2006. (Cited on page 83.)

[8] Victor R. Basili. Software modeling and measurement: the Goal/Question/Metric paradigm. Technical report, University of Maryland, 1992. (Cited on page 14.)

[9] Victor R. Basili. The experimental paradigm in software engineering. In *Experimental Software Engineering Issues: Critical Assessment and Future Directions*, Lecture Notes in Computer Science, pages 1–12. Springer Berlin Heidelberg, 1 January 1993. (Cited on page 6.)

[10] Victor R. Basili. A personal perspective on the evolution of empirical software engineering. In Jürgen Münch and Klaus Schmid, editors, *Perspectives on the Future of Software Engineering*, pages 255–273. Springer Berlin Heidelberg, 2013. `doi:10.1007/978-3-642-37395-4_17`. (Cited on page 119.)

[11] Victor R. Basili, Richard W. Selby, and David H. Hutchens. Experimentation in software engineering. *IEEE Trans. Software Eng.*, SE-12(7):733–743, July 1986. (Cited on page 38.)

[12] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. SEI Series in Software Engineering. Addison-Wesley, third edition, 2012. (Cited on pages 5, 69, 70, 87, and 89.)

[13] Andreas Berl, Erol Gelenbe, Marco Di Girolamo, Giovanni Giuliani, Hermann De Meer, Minh Quan Dang, and Kostas Pentikousis. Energy-efficient cloud computing. *The Computer Journal*, 53(7):1045–1051, 2010. (Cited on page 65.)

[14] Mathias Binswanger. Technological progress and sustainable development: what about the rebound effect? *Ecol. Econ.*, 36(1):119–132, January 2001. (Cited on page 3.)

[15] Paolo Bozzelli, Qing Gu, and Patricia Lago. A systematic literature review on green software metrics. Technical report, VU University Amsterdam, 2013. (Cited on page 66.)

[16] Antonio Brogi, Ahmad Ibrahim, Jacopo Soldani, José Carrasco, Javier Cubo, Ernesto Pimentel, and Francesco D'Andria. SeaClouds: A European project on seamless management of multi-cloud applications. *SIGSOFT Softw. Eng. Notes*, 39(1):1–4, February 2014. (Cited on page 76.)

[17] Kirk W. Cameron. Energy oddities, part 2: Why green computing is odd. *Computer*, 46(3):90–93, 2013. `doi:10.1109/MC.2013.94`. (Cited on page 114.)

[18] Eugenio Capra, Chiara Francalanci, and Sandra A. Slaughter. Is software "green"? Application development environments and energy efficiency in open source applications. *Information and Software Technology*, 54(1):60–71, January 2012. `doi:10.1016/j.infsof.2011.07.005`. (Cited on pages 39, 40, and 114.)

[19] Eugenio Capra, Chiara Francalanci, and Sandra A. Slaughter. Measuring application software energy efficiency. *IT Professional*, 14(2):54–61, March/April 2012. `doi:10.1109/MITP.2012.39`. (Cited on page 114.)

[20] Eugenio Capra and Francesco Merlo. Green it: Everything starts from the software. In Susan Newell, Edgar A. Whitley, Nancy Pouloudi, Jonathan Wareham, and Lars Mathiassen, editors, *17th European Conference on Information Systems*, pages 62–73, Verona, Italy, 2009. (Cited on page 86.)

[21] Julien Carpentier, Jean-Patrick Gelas, Laurent Lefevre, Maxime Morel, Olivier Mornard, and Jean-Pierre Laisne. Compatibleone: Designing an energy efficient open source cloud broker. In *Cloud and Green Computing (CGC), 2012 Second International Conference on*, pages 199–205. IEEE, 2012. (Cited on pages 73, 78, 81, 82, and 90.)

[22] Pablo Jesus Chacin Martìnez et al. *A Middleware framework for self-adaptive large scale distributed services.* PhD thesis, Universitat Politecnica de Catalunya, Departament d'Arquitectura dels Computadors, 2011. (Cited on pages 72, 78, 81, and 82.)

[23] FeiFei Chen, J Schneider, Yun Yang, John Grundy, and Qiang He. An energy consumption model and analysis tool for cloud computing environments. In *First International Workshop on Green and Sustainable Software (GREENS)*, pages 45–50. IEEE, 2012. (Cited on page 35.)

[24] Thomas D. Cook and Donald T. Campbell. *Quasi-experimentation: Design & analysis issues for field settings.* Houghton Mifflin Company, Boston, 1979. (Cited on page 52.)

[25] Intel Corp. Intel Energy Checker SDK, 2010. Available from: `https://software.intel.com/en-us/articles/intel-energy-checker-sdk`. (Cited on page 48.)

[26] Lewis Curtis. Environmentally sustainable infrastructure design. *The Architecture Journal*, 18:2–8, 2008. (Cited on pages 72, 78, 81, 82, and 109.)

[27] Remco C. De Boer, Rik Farenhorst, Patricia Lago, Hans Van Vliet, Viktor Clerc, and Anton Jansen. Architectural knowledge: Getting to the core. In *Software Architectures, Components, and Applications*, pages 197–214. Springer, 2007. (Cited on page 69.)

[28] Frederico G. Alvares De Oliveira Jr, Thomas Ledoux, et al. Self-optimisation of the energy footprint in service-oriented architectures. In *Proceedings of the 1^{st} Workshop on Green Computing*, pages 4–9, 2010. (Cited on pages 72, 78, 81, 82, and 93.)

[29] Yadolah Dodge. *The Oxford dictionary of statistical terms.* Oxford University Press, 2006. (Cited on page 32.)

[30] Brian Patrick Dougherty. *Configuration and Deployment Derivation Strategies for Distributed Real-time and Embedded Systems.* PhD thesis, Vanderbilt University, 2011. (Cited on pages 72, 78, 79, 81, and 82.)

[31] Corentin Dupont, Giovanni Giuliani, Fabieu Hermenier, Thomas Schulze, and Andrev Somov. An energy aware framework for virtual machine placement in cloud federated data centres. In *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*, pages 1–10. IEEE, 2012. (Cited on pages 73, 78, 81, 82, and 95.)

[32] Thomas Erl. *SOA: principles of service design*, volume 1. Prentice Hall, 2008. (Cited on page 97.)

[33] Niroshinie Fernando, Seng W Loke, and Wenny Rahayu. Mobile cloud computing: A survey. *Future Generation Computer Systems*, 29(1):84–106, 2013. (Cited on pages 36 and 86.)

[34] Miguel Ferreira, Eric Hoekstra, Bo Merkus, and Joost Visser. Seflab: A lab for measuring software energy footprints. In *Proceedings of the Second International Workshop on Green and Sustainable Software (GREENS)*, 2013. (Cited on pages 32, 38, 47, and 118.)

[35] N Ferry, A Rossini, F Chauvel, B Morin, and others. Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems. *CLOUD 2013: IEEE*, 2013. (Cited on page 76.)

[36] Tim Forell, Dejan Milojicic, and Vanish Talwar. Cloud management: Challenges and opportunities. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 881–889. IEEE, 2011. (Cited on pages 72, 78, 81, and 82.)

[37] Marting Fowler and Kent Beck. *Refactoring: improving the design of existing code.* Addison-Wesley Professional, 1999. (Cited on page 109.)

[38] E Gamma, R Helm, R Johnson, and J Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Pearson Education, 1994. (Cited on page 41.)

[39] Saurabh Kumar Garg, Chee Shin Yeo, and Rajkumar Buyya. Green cloud framework for improving carbon efficiency of clouds. In *Euro-Par 2011 Parallel Processing*, Lecture Notes in Computer Science, pages 491–502. Springer Berlin Heidelberg, 1 January 2011. (Cited on pages 72, 78, 81, 82, and 98.)

[40] David Garlan and Mary Shaw. An introduction to software architecture. *Advances in software engineering and knowledge engineering*, 1:1–40, 1993. (Cited on page 70.)

[41] Mark Gordon, Lide Zhang, and Birjodh Tiwana. Powertutor, 2009. Last visited: May 5th, 2014. Available from: `http://ziyang.eecs.umich.edu/projects/powertutor/`. (Cited on page 108.)

[42] Marion Gottschalk, Mirco Josefiok, Jan Jelschen, and Andreas Winter. Removing energy code smells with reengineering services. In *GI-Jahrestagung*, pages 441–455, 2012. (Cited on page 109.)

[43] Sebastian Götz, Claas Wilke, Sebastian Cech, and Uwe Assmann. Runtime variability management for energy-efficient software by contract negotiation. In *Proceedings of the 6^{th} International Workshop Models@run.time (MRT 2011)*, 2011. (Cited on pages 72, 78, 81, and 82.)

[44] Greenpeace. Make it green: Cloud computing and its contribution to climate change. Technical report, Greenpeace International, 30 March 2010. (Cited on page 1.)

[45] Qing Gu, Patricia Lago, and Simone Potenza. Delegating data management to the cloud: a case study in a telecommunication company. In *International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA)*, volume 7, pages 56–63. IEEE Computer Society, sep 2013. (Cited on pages 85 and 86.)

[46] Sven Gude. Energy efficient software. Master's thesis, VU University Amsterdam, Sep 2010. (Cited on pages 44, 45, and 61.)

[47] PK Gupta and G Singh. Minimizing power consumption by personal computers: A technical survey. *International Journal of Information Technology and Computer Science*, 4(10):57, 2012. (Cited on page 32.)

[48] Shuai Hao, Ding Li, William GJ Halfond, and Ramesh Govindan. Estimating mobile application energy consumption using program analysis. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 92–101. IEEE Press, 2013. (Cited on page 34.)

[49] Mark Harman, Kiran Lakhotiaa, Jeremy Singerb, David R Whiteb, and Shin Yooa. Cloud engineering is search based optimization too. *Journal of Systems and Software*, 86(9):2225–2241, 2013. (Cited on pages 73, 78, 81, and 82.)

[50] Markus Hedwig. *Taming energy costs of large enterprise systems through adaptive provisioning.* PhD thesis, Albert-Ludwigs-Universität Freiburg, 2009. (Cited on pages 72, 78, 81, and 82.)

[51] Nikolas Roman Herbst, Samuel Kounev, and Ralf Reussner. Elasticity in cloud computing: What it is, and what it is not. In *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*, pages 23–27, San Jose, CA, 2013. USENIX. Available from: `https://www.usenix.org/conference/icac13/technical-sessions/presentation/herbst`. (Cited on page 96.)

[52] Abram Hindle. Green mining: a methodology of relating software change and configuration to power consumption. *Empir. Softw. Eng.*, pages 1–36, 2013. (Cited on pages 40 and 41.)

[53] Nikolaus Huber, Fabian Brosig, and Samuel Kounev. Model-based self-adaptive resource allocation in virtualized environments. In *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 90–99. ACM, 2011. (Cited on pages 72, 78, 81, and 82.)

[54] Intel Open Source Technology Center. Powertop, 2012. Last visited: May 5th, 2014. Available from: `https://01.org/powertop`. (Cited on page 108.)

[55] ISO/IEC/IEEE. Systems and software engineering - architecture description, mar 2011. Available from: `http://www.iso-architecture.org/ieee-1471/`. (Cited on pages 69 and 107.)

[56] Nick Jones. Eight software approaches can enable Energy-Efficient computing. Technical Report G00151775, Gartner, 16 October 2007. (Cited on page 44.)

[57] Vigdis By Kampenes, Tore Dybå, Jo E Hannay, and Dag I K. Sjøberg. A systematic review of quasi-experiments in software engineering. *Information and Software Technology*, 51(1):71–82, January 2009. (Cited on page 7.)

[58] Aman Kansal and Feng Zhao. Fine-grained energy profiling for power-aware application design. *SIGMETRICS Perform. Eval. Rev.*, 36(2):26–31, August 2008. `doi:10.1145/1453175.1453180`. (Cited on page 37.)

[59] Gregory Katsaros, Josep Subirats, J Oriol Fitó, Jordi Guitart, Pierre Gilet, and Daniel Espling. A service framework for energy-aware monitoring

and vm management in clouds. *Future Generation Computer Systems*, InPress:InPress, 2012. (Cited on pages 73, 78, 81, and 82.)

[60] R Kazman, J Asundi, and M Klein. Quantifying the costs and benefits of architectural decisions. In *Software Engineering, 2001. ICSE 2001. Proceedings of the 23rd International Conference on*, pages 297–306, May 2001. (Cited on page 70.)

[61] J Kephart, JO Kephart, DM Chess, Craig Boutilier, Rajarshi Das, Jeffrey O Kephart, and William E Walsh. An architectural blueprint for autonomic computing. Technical report, IBM Corporation, 2005. 3rd Ed. (Cited on page 100.)

[62] Nakku Kim, Jungwook Cho, and Euiseong Seo. Energy-credit scheduler: an energy-aware virtual machine scheduler for cloud systems. *Future Generation Computer Systems*, 2012. (Cited on page 34.)

[63] Barbara Kitchenham, O Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology*, 51(1):7–15, 2009. (Cited on page 66.)

[64] Barbara A Kitchenham and Stuart Charters. Guidelines for performing systematic literature reviews in software engineering (version 2.3). Technical Report EBSE-2007-01, EBSE, 2007. (Cited on pages 6 and 69.)

[65] J G Koomey, S Berard, M Sanchez, and H Wong. Implications of historical trends in the electrical efficiency of computing. *IEEE Ann. Hist. Comput.*, 33(3):46–54, March 2011. (Cited on page 3.)

[66] Jonathan Koomey. Growth in data center electricity use 2005 to 2010. *Oakland, CA: Analytics Press. August*, 1:2010, 2011. (Cited on page 1.)

[67] Samuel Kounev. Self-aware software and systems engineering: A vision and research roadmap. *GI Softwaretechnik-Trends*, 31 (4):21–25, 2011. (Cited on pages 72, 78, 81, and 82.)

[68] Tobias Kurze, Markus Klems, David Bermbach, Alexander Lenk, Stefan Tai, and Marcel Kunze. Cloud federation. In *The Second International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING)*, pages 32–38, 2011. (Cited on page 97.)

[69] Young-Woo Kwon and Eli Tilevich. The impact of distributed programming abstractions on application energy consumption. *Information and Software Technology*, 55(9):1602–1613, September 2013. (Cited on pages 40 and 41.)

[70] Patricia Lago. A master program on engineering energy-aware software. In *Proceedings of ICT for Energy Efficiency (EnviroInfo)*, 2014. (Cited on page 125.)

[71] Patricia Lago, Rick Kazman, Niklaus Meyer, Maurizio Morisio, Hausi A. Müller, Frances Paulisch, Giuseppe Scanniello, Birgit Penzenstadler, and Olaf Zimmermann. Exploring initial challenges for green software engineering: summary of the first GREENS workshop, at ICSE 2012. *SIGSOFT Softw. Eng. Notes*, 38(1):31–33, January 2013. `doi:10.1145/2413038.2413062`. (Cited on pages 107, 113, and 114.)

[72] Patricia Lago, Sedef Akinli Kocak, Ivica Crnkovic, Henning Femmer, Hausi A. Müller, and Birgit Penzenstadler. Framing sustainability as quality property of green software. *Submitted to Communications of the ACM*, 2014. Under review. (Cited on page 125.)

[73] Patricia Lago, Niklaus Meyer, Maurizio Morisio, Hausi Muller, and Giuseppe Scaniello. 2nd international workshop on green and sustainable software (greens 2013). In *Proceedings of the 2013 International Conference on Software Engineering: Companion Volume*, pages 1545–1546. IEEE Computer Society, may 2013. (Cited on pages 84 and 107.)

[74] Patricia Lago, Niklaus Meyer, Maurizio Morisio, Hausi A. Müller, and Giuseppe Scanniello. Leveraging "Energy Efficiency to Software Users" : summary of the Second GREENS workshop, at ICSE 2013. *SIGSOFT Softw. Eng. Notes*, January 2013. `doi:10.1145/2413038.2413062`. (Cited on page 2.)

[75] John A. Laitner and Mike Berners-Lee. Gesi smarter 2020: The role of ICT in driving a sustainable future. Technical report, Global e-Sustainability Initiative, 2012. (Cited on page 1.)

[76] Bart Lannoo. Overview of ICT energy consumption. Technical Report D8.1, iMinds, 2 May 2013. Deliverable of the EU FP7 Project "Network of Excellence in Internet Science" (EINS). (Cited on page 2.)

[77] P Larsson. Energy-efficient software guidelines. *Intel Software Solutions Group, Tech. Rep*, 2011. (Cited on pages 37 and 44.)

[78] D Li and Wgj Halfond. An investigation into energy-saving programming practices for android smartphone app development. *Proceedings of the 3rd International Workshop on Green and Sustainable Software (GREENS)*, 2014. (Cited on pages 40 and 42.)

[79] M Linares-Vásquez, G Bavota, C Bernal-Cárdenas, and others. Mining Energy-Greedy API usage patterns in android apps: An empirical study. In *11th Working Conference on Mining Software Repositories*. ACM New York, NY, USA, 2014. (Cited on pages 40 and 43.)

[80] Lanyue Lu, Peter J Varman, and Kshitij Doshi. Decomposing workload bursts for efficient storage resource management. *Parallel and Distributed Systems, IEEE Transactions on*, 22(5):860–873, 2011. (Cited on pages 72, 78, 81, 82, and 97.)

[81] M. Marcu, M. Vladutiu, H. Moldovan, and M. Popa. Thermal benchmark and power benchmark software. *ArXiv e-prints*, September 2007. `arXiv:0709.1834`. (Cited on page 13.)

[82] Erix Masanet, Arman Shehabi, Lavanya Ramakrishnan, Jiaqi Liang, Xiaohui Ma, Benjamin Walker, Valerie Hendrix, and Pradeep Maantha. The energy efficiency potential of cloud-based software: A u.s. case study. Technical report, Laurence Berkeley National Lab, Berkeley, California, June 2013. (Cited on pages 85 and 86.)

[83] Microsoft Research. Joulemeter, 2008. Last visited: May 5th, 2014. Available from: `http://research.microsoft.com/en-us/projects/joulemeter/`. (Cited on page 108.)

[84] Matthew B Miles and A Michael Huberman. *Qualitative data analysis: An expanded sourcebook*. Sage, 1994. (Cited on page 69.)

[85] John A Mills. A pragmatic view of the system architect. *Commun. ACM*, 28(7):708–717, July 1985. (Cited on pages 82 and 107.)

[86] D C Montgomery. *Design and Analysis of Experiments*. Student solutions manual. John Wiley & Sons, 2008. (Cited on page 51.)

[87] G E Moore. Cramming more components onto integrated circuits. *Electronics,*, 38(8), April 1965. (Cited on page 2.)

[88] Davide Morelli, Antonio Cisternino, and Andrew V Jones. A compositional model to characterize software and hardware from their resource usage. In *Proceedings of Imperial College Computing Student Workshop (ICCSW)*, pages 95–101, 2012. (Cited on page 34.)

[89] Adel Noureddine, Aurelien Bourdon, Romain Rouvoy, and Lionel Seinturier. A preliminary study of the impact of software engineering on greenit. In *2012 First International Workshop on Green and Sustainable Software (GREENS)*, pages 21–27. IEEE, 2012. (Cited on pages 40 and 41.)

[90] Adel Noureddine, Romain Rouvoy, and Lionel Seinturier. Supporting energy-driven adaptations in distributed environments. In *Proceedings of the 1st Workshop on Middleware and Architectures for Autonomic and Sustainable Computing*, pages 13–18. ACM, 2011. (Cited on pages 72, 78, 81, and 82.)

[91] Adel Noureddine, Romain Rouvoy, and Lionel Seinturier. A review of middleware approaches for energy management in distributed environments. *Software: Practice and Experience*, 00:1–30, 2012. (Cited on pages 73, 78, 81, and 82.)

[92] Adam Oliner, Anand Padmanabha Iyer, Ion Stoica, Eemil Lagerspetz, and Sasu Tarkoma. Carat: Collaborative energy diagnosis for mobile devices. Technical report, University of California at Berkeley, 2013. (Cited on page 35.)

[93] Rajesh Palit, Ajit Singh, and Kshirasagar Naik. Energy cost of software applications on portable wireless devices. In *Green Mobile Devices and Networks: Energy Optimization and Scavenging Techniques*, page 53. CRC Press, 2012. (Cited on pages 34, 35, and 36.)

[94] Abhinav Pathak, Y Charlie Hu, and Ming Zhang. Bootstrapping energy debugging on smartphones: A first look at energy bugs in mobile devices. In *10th ACM Workshop on Hot Topics in Networks*, HotNets-X, pages 5:1–5:6, New York, NY, USA, 2011. ACM. (Cited on pages 109 and 115.)

[95] Birgit Penzenstadler, Henning Femmer, and Debra Richardson. Who is the Advocate? Stakeholders for Sustainability. In *2nd International Workshop on Green and Sustainable Software (GREENS) at ICSE*, 2013. (Cited on page 107.)

[96] Dewayne E Perry and Alexander L Wolf. Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, 17(4):40–52, 1992. (Cited on page 69.)

[97] G Pinto, F Castor, and Y D Liu. Understanding energy behaviors of thread management constructs. In *28th ACM Conference on Object-Oriented programming systems, languages, and applications (OOPSLA)*, 2014. (Cited on pages 40 and 42.)

[98] Gustavo Pinto, Fernando Castor, and Yu David Liu. Mining questions about software energy consumption. In *The 11th Working Conference on Mining Software Repositories (MSR)*, 2014. (Cited on pages 40 and 43.)

[99] Gerald J Popek and Robert P Goldberg. Formal requirements for virtualizable third generation architectures. *Communications of the ACM*, 17(7):412–421, 1974. (Cited on page 94.)

[100] W Pree. Meta patterns —a means for capturing the essentials of reusable object-oriented design. *Object-oriented programming*, 1994. (Cited on page 115.)

[101] Claudia Raibulet and Laura Masciadri. Evaluation of dynamic adaptivity through metrics: an achievable target? In *Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on*, pages 341–344. IEEE, 2009. (Cited on page 111.)

[102] Parthasarathy Ranganathan, Suzanne Rivoire, and Justin Moore. Chapter 3 models and metrics for energy-efficient computing. In Marvin V. Zelkowitz, editor, *Computer Performance Issues*, volume 75 of *Advances in Computers*, pages 159–233. Elsevier, 2009. Available from: `http://www.sciencedirect.com/science/article/B7RNF-4W1SCVX-6/2/e125a98e7a8d0f85f3bc2da984764d20`, `doi:DOI:10.1016/S0065-2458(08)00803-6`. (Cited on page 33.)

[103] Govindaraj Rangaraj and Rami Bahsoon. Green software architectures: A market-based approach. In *The Second International Workshop on Software Research and Climate Change (WSRCC), in affiliation with the ACM/IEEE 32nd International Conference on Software Engineering (ICSE)*, ICSE'10, 2010. (Cited on page 86.)

[104] Sherief Reda and Abdullah N. Nowroz. Power modeling and characterization of computing devices: A survey. *Found. Trends Electron. Des. Autom.*, 6(2):121–216, February 2012. `doi:10.1561/1000000022`. (Cited on pages 32, 33, and 92.)

[105] Suzanne Rivoire, Mehul A. Shah, Parthasarathy Ranganathan, Christos Kozyrakis, and Justin Meza. Models and metrics to enable energy-efficiency optimizations. *Computer*, 40(12):39–48, 2007. `doi:10.1109/MC.2007.436`. (Cited on page 33.)

[106] Dan Rogers and Ulrich Homann. Application patterns for green it. *The Architecture Journal*, 18:16–21, 2008. (Cited on pages 72, 78, 81, 82, and 94.)

[107] C Sahin, F Cayci, I L M Gutierrez, J Clause, F Kiamilev, L Pollock, and K Winbladh. Initial explorations on design pattern energy usage. In *Green and Sustainable Software (GREENS), 2012 First International Workshop on*, pages 55–61, June 2012. (Cited on page 40.)

[108] J Saldana. *The Coding Manual for Qualitative Researchers*. English short title catalogue Eighteenth Century collection. SAGE Publications, 2012. (Cited on page 69.)

[109] M. Satyanarayanan. Pervasive computing: vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, 2001. `doi:10.1109/98.943998`. (Cited on page 86.)

[110] E Saxe. Power-Efficient software. *Commun. ACM*, 2010. (Cited on pages 44 and 114.)

[111] Simon Schubert, Dejan Kostic, Willy Zwaenepoel, and Kang G Shin. Profiling software for energy consumption. In *IEEE International Conference on Green Computing and Communications*, pages 515–522. IEEE, 2012. (Cited on pages 34 and 36.)

[112] Jyothi Sekhar, Getzi Jeba, and S Durga. A survey on energy efficient server consolidation through vm live migration. *International Journal of Advances in Engineering & Technology*, 5 (1):515–525, 2012. (Cited on pages 73, 78, 81, and 82.)

[113] Chiyoung Seo, George Edwards, Daniel Popescu, Sam Malek, and Nenad Medvidovic. A framework for estimating the energy consumption induced by a distributed system's architectural style. In *Proceedings of the 8th international workshop on Specification and verification of component-based systems*, SAVCBS '09, pages 27–34, New York, NY, USA, 2009. ACM. `doi:10.1145/1596486.1596493`. (Cited on page 86.)

[114] Chiyoung Seo, Sam Malek, and Nenad Medvidovic. Component-level energy consumption estimation for distributed java-based software systems. In *Component-Based Software Engineering*, pages 97–113. Springer, 2008. (Cited on page 110.)

[115] M Sevalnev, S Aalto, J Kommeri, and T Niemi. Using queuing theory for controlling the number of computing servers. In *Proceedings of the 3rd International Conference on Green IT Solutions (ICGREEN 2012)*. SciTePress, July 2012. (Cited on pages 73, 78, 81, and 82.)

[116] S S Shapiro and M B Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1 December 1965. (Cited on page 51.)

[117] Junaid Shuja, SajjadA. Madani, Kashif Bilal, Khizar Hayat, SameeU. Khan, and Shahzad Sarwar. Energy-efficient data centers. *Computing*, 94(12):973–994, 2012. `doi:10.1007/s00607-012-0211-2`. (Cited on page 35.)

[118] T Simunic, L Benini, and G De Micheli. Energy-efficient design of battery-powered embedded systems. *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, 9(1):15–28, February 2001. (Cited on page 44.)

[119] Amit Sinha and Anantha P Chandrakasan. Energy aware software. In *Thirteenth International Conference on VLSI Design*, pages 50–55. IEEE, 2000. (Cited on pages 32, 34, and 37.)

[120] David Mitchell Smith. Hype cycle for cloud computing 2011. *Gartner Inc., Stamford*, 2011. (Cited on pages 71 and 74.)

[121] William J Song, Sudhakar Yalamanchili, Arun F Rodrigues, and Saibal Mukhopadhyay. Instruction-based energy estimation methodology for asymmetric manycore processor simulations. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*, pages 166–171, 2012. (Cited on page 34.)

[122] B Steigerwald, R Chabukswar, K Krishnan, and JD Vega. Creating Energy-Efficient software. Technical report, Intel Corp., 2007. (Cited on pages 44 and 109.)

[123] T H Szymanski. Impact of future trends on exascale grid and cloud computing. In *Supercomputing*, pages 215–231. Springer International Publishing, Jan 2014. (Cited on page 61.)

[124] Steven te Brinke, Somayeh Malakuti, Christoph Bockisch, Lodewijk Bergmans, and Mehmet Akşit. A design method for modular energy-aware software. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1180–1182, New York, NY, USA, 2013. ACM. `doi:10.1145/2480362.2480584`. (Cited on page 86.)

[125] TERNA. Dati Statistici sull'energia elettrica in Italia. Technical report, SISTAN, 2013. (Cited on page 38.)

[126] Vivek Tiwari, Sharad Malik, and Andrew Wolfe. Power analysis of embedded software: a first step towards software power minimization. *IEEE Transactions on Very Large Scale Integration Systems*, 2(4):437–445, 1994. (Cited on pages 32 and 34.)

[127] N Tolia, Z Wang, M Marwah, C Bash, P Ranganathan, and others. Delivering energy proportionality with non Energy-Proportional Systems-Optimizing the ensemble. *HotPower*, 2008. (Cited on page 3.)

[128] Shiao-Li Tsao and Jian Jhen Chen. Seprof: A high-level software energy profiling tool for an embedded processor enabling power management functions. *Journal of Systems and Software*, 85(8):1757–1769, 2012. (Cited on page 34.)

[129] A Tzanakaki, M Anastasopoulos, K Georgakilas, J Buysse, M De Leenheer, C Develder, Shuping Peng, R Nejabati, E Escalona, D Simeonidou, N Ciulli, G Landi, M Brogle, A Manfredi, E Lopez, J F Riera, J A Garcia-Espin, P Donadio, G Parladori, and J Jimenez. Energy efficiency in integrated IT and optical network infrastructures: The GEYSERS approach. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 343–348. ieeexplore.ieee.org, April 2011. (Cited on pages 72, 78, 81, 82, and 92.)

[130] András Vargha and Harold D Delaney. A critique and improvement of the "CL" common language effect size statistics of McGraw and wong. *J. Educ. Behav. Stat.*, 25(2):101–132, 1 July 2000. (Cited on page 51.)

[131] A Vetrò, L Ardito, G Procaccianti, and M Morisio. Definition, implementation and validation of energy code smells: an exploratory study on an embedded system. In *ENERGY 2013 : The Third International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, pages 34–39, 2013. (Cited on pages 104, 109, and 115.)

[132] David Villegas, Norman Bobroff, Ivan Rodero, Javier Delgado, Yanbin Liu, Aditya Devarakonda, Liana Fong, S Masoud Sadjadi, and Manish Parashar. Cloud federation in a layered service model. *Journal of Computer and System Sciences*, 78(5):1330–1344, 2012. (Cited on pages 73, 78, 81, 82, and 99.)

[133] Roel J. Wieringa. *Design Science Methodology for Information Systems and Software Engineering.* Springer, September 2014. (Cited on pages 45 and 117.)

[134] Claas Wilke, Sebastian Gotz, and Sebastian Richly. Jouleunit: a generic framework for software energy profiling and testing. In *Proceedings of the 2013 workshop on Green in/by software engineering*, pages 9–14. ACM, 2013. (Cited on page 32.)

[135] LG Williams and CU Smith. Five steps to solving software performance problems. *Software Engineering Research and Performance Engineering Services*, 2002. (Cited on page 115.)

[136] Niklaus Wirth. A plea for lean software. *Computer*, 28(2):64–68, February 1995. (Cited on page 2.)

[137] C. Wohlin. *Experimentation in Software Engineering: An Introduction*. The Kluwer International Series in Software Engineering. Kluwer Academic, 2000. Available from: `http://books.google.es/books?id=nG2UShV0wAEC`. (Cited on pages 83 and 116.)

[138] Claes Wohlin, Per Runeson, Martin Hst, Magnus C Ohlsson, Bjrn Regnell, and Anders Wessln. *Experimentation in software engineering*. Springer Publishing Company, Incorporated, 2012. (Cited on pages 7, 38, 39, and 44.)

[139] Zhengkai Wu, Michael E Cotterell, Sun Qin, Aaron Beach, and Grijalva Santiago. Towards a cloud infrastructure for energy informatics. In *Energy Informatics. Sprouts: Working Papers on Information Systems*, 2012. (Cited on pages 73, 78, 81, and 82.)

[140] N Xiong, W Han, and A Vandenberg. Green cloud computing schemes based on networks: a survey. *Communications, IET*, 6(18):3294–3300, 2012. (Cited on pages 73, 78, 81, and 82.)

[141] Li Xu, Guozhen Tan, Xia Zhang, and Jingang Zhou. Energy aware cloud application management in private cloud data center. In *Cloud and Service Computing (CSC), 2011 International Conference on*, pages 274–279. IEEE, 2011. (Cited on pages 72, 78, 81, 82, and 94.)

[142] Li Xu, Guozhen Tan, Xia Zhang, and Jingang Zhou. A bdi agent-based approach for cloud application autonomic management. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4^{th} International Conference on*, pages 574–577. IEEE, 2012. (Cited on pages 73, 78, 81, and 82.)

[143] Paula Younger. Using google scholar to conduct a literature search. *Nursing Standard*, 24(45):40–46, 2010. (Cited on page 83.)

[144] Hang Yuan, C.-C. Jay Kuo, and Ishfaq Ahmad. Energy efficiency in data centers and cloud-based multimedia services: An overview and future directions. *International Conference on Green Computing*, 0:375–382, 2010. `doi:10.1109/GREENCOMP.2010.5598292`. (Cited on page 35.)

[145] Z. Zhang and J. Chang. A cool scheduler for multi-core systems exploiting program phases. *IEEE Transactions on Computers*, PP(99):1–1, 2012. `doi:10.1109/TC.2012.283`. (Cited on page 34.)

[146] Albert Y Zomaya and Young Choon Lee. *Energy efficient distributed computing systems*, volume 88. Wiley, 2012. (Cited on pages 32 and 33.)

SIKS Dissertation Series

1998

1998-1 Johan van den Akker (CWI)
DEGAS - An Active, Temporal Database of Autonomous Objects

1998-2 Floris Wiesman (UM)
Information Retrieval by Graphically Browsing Meta-Information

1998-3 Ans Steuten (TUD)
A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective

1998-4 Dennis Breuker (UM)
Memory versus Search in Games

1998-5 E.W.Oskamp (RUL)
Computerondersteuning bij Straftoemeting

1999

1999-1 Mark Sloof (VU)
Physiology of Quality Change Modelling;
Automated modelling of Quality Change of Agricultural Products

1999-2 Rob Potharst (EUR)
Classification using decision trees and neural nets

1999-3 Don Beal (UM)
The Nature of Minimax Search

1999-4 Jacques Penders (UM)
The practical Art of Moving Physical Objects

1999-5 Aldo de Moor (KUB)
Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems

1999-6 Niek J.E. Wijngaards (VU)
Re-design of compositional systems

1999-7 David Spelt (UT)
Verification support for object database design

1999-8 Jacques H.J. Lenting (UM)
Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.

2000

2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance

2000-2 Koen Holtman (TUE)
Prototyping of CMS Storage Management

2000-3 Carolien M.T. Metselaar (UVA)
Sociaal-organisatorische gevolgen van kennistechnologie;
een procesbenadering en actorperspectief.

2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface Design

2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval.

2000-6 Rogier van Eijk (UU)
Programming Languages for Agent Communication

2000-7 Niels Peek (UU)
Decision-theoretic Planning of Clinical Patient Management

2000-8 Veerle Coup (EUR)
Sensitivity Analyis of Decision-Theoretic Networks

2000-9 Florian Waas (CWI)
Principles of Probabilistic Query Optimization

2000-10 Niels Nes (CWI)
Image Database Management System Design Considerations,
Algorithms and Architecture

2000-11 Jonas Karlsson (CWI)
Scalable Distributed Data Structures for Database Management

2001

2001-1 Silja Renooij (UU)
Qualitative Approaches to Quantifying Probabilistic Networks

2001-2 Koen Hindriks (UU)
Agent Programming Languages: Programming with Mental Models

2001-3 Maarten van Someren (UvA)
Learning as problem solving

2001-4 Evgueni Smirnov (UM)
Conjunctive and Disjunctive Version Spaces with
Instance-Based Boundary Sets

2001-5 Jacco van Ossenbruggen (VU)
Processing Structured Hypermedia: A Matter of Style

2001-6 Martijn van Welie (VU)
Task-based User Interface Design

2001-7 Bastiaan Schonhage (VU)
Diva: Architectural Perspectives on Information Visualization

2001-8 Pascal van Eck (VU)
A Compositional Semantic Structure for Multi-Agent Systems Dynamics.

2001-9 Pieter Jan 't Hoen (RUL)
Towards Distributed Development of Large Object-Oriented Models,
Views of Packages as Classes

2001-10 Maarten Sierhuis (UvA)
Modeling and Simulating Work Practice
BRAHMS: a multiagent modeling and simulation language
for work practice analysis and design

2001-11 Tom M. van Engers (VUA)
Knowledge Management:
The Role of Mental Models in Business Systems Design

2002

2002-01 Nico Lassing (VU)
Architecture-Level Modifiability Analysis

2002-02 Roelof van Zwol (UT)
Modelling and searching web-based document collections

2002-03 Henk Ernst Blok (UT)
Database Optimization Aspects for Information Retrieval

2002-04 Juan Roberto Castelo Valdueza (UU)
The Discrete Acyclic Digraph Markov Model in Data Mining

2002-05 Radu Serban (VU)
The Private Cyberspace Modeling Electronic Environments
inhabited by Privacy-concerned Agents

2002-06 Laurens Mommers (UL)
Applied legal epistemology;
Building a knowledge-based ontology of the legal domain

2002-07 Peter Boncz (CWI)
Monet: A Next-Generation DBMS Kernel For Query-Intensive
Applications

2002-08 Jaap Gordijn (VU)
Value Based Requirements Engineering: Exploring Innovative
E-Commerce Ideas

2002-09 Willem-Jan van den Heuvel(KUB)
Integrating Modern Business Applications with Objectified
Legacy Systems

2002-10 Brian Sheppard (UM)
Towards Perfect Play of Scrabble

2002-11 Wouter C.A. Wijngaards (VU)
Agent Based Modelling of Dynamics: Biological and Organisational
Applications

2002-12 Albrecht Schmidt (Uva)
Processing XML in Database Systems

2002-13 Hongjing Wu (TUE)
A Reference Architecture for Adaptive Hypermedia Applications

2002-14 Wieke de Vries (UU)
Agent Interaction: Abstract Approaches to Modelling,
Programming and
Verifying Multi-Agent Systems

2002-15 Rik Eshuis (UT)
Semantics and Verification of UML Activity Diagrams for
Workflow Modelling

2002-16 Pieter van Langen (VU)

The Anatomy of Design: Foundations, Models and Applications

2002-17 Stefan Manegold (UVA)
Understanding, Modeling, and Improving Main-Memory Database Performance

2003

2003-01 Heiner Stuckenschmidt (VU)
Ontology-Based Information Sharing in Weakly Structured Environments

2003-02 Jan Broersen (VU)
Modal Action Logics for Reasoning About Reactive Systems

2003-03 Martijn Schuemie (TUD)
Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy

2003-04 Milan Petkovic (UT)
Content-Based Video Retrieval Supported by Database Technology

2003-05 Jos Lehmann (UVA)
Causation in Artificial Intelligence and Law - A modelling approach

2003-06 Boris van Schooten (UT)
Development and specification of virtual environments

2003-07 Machiel Jansen (UvA)
Formal Explorations of Knowledge Intensive Tasks

2003-08 Yongping Ran (UM)
Repair Based Scheduling

2003-09 Rens Kortmann (UM)
The resolution of visually guided behaviour

2003-10 Andreas Lincke (UvT)
Electronic Business Negotiation: Some experimental studies on the interaction
between medium, innovation context and culture

2003-11 Simon Keizer (UT)
Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks

2003-12 Roeland Ordelman (UT)
Dutch speech recognition in multimedia information retrieval

2003-13 Jeroen Donkers (UM)
Nosce Hostem - Searching with Opponent Models

2003-14 Stijn Hoppenbrouwers (KUN)
Freezing Language: Conceptualisation Processes across ICT-Supported Organisations

2003-15 Mathijs de Weerdt (TUD)
Plan Merging in Multi-Agent Systems

2003-16 Menzo Windhouwer (CWI)
Feature Grammar Systems - Incremental Maintenance of Indexes to
Digital Media Warehouses

2003-17 David Jansen (UT)
Extensions of Statecharts with Probability, Time, and Stochastic Timing

2003-18 Levente Kocsis (UM)

Learning Search Decisions

2004

2004-01 Virginia Dignum (UU)
A Model for Organizational Interaction: Based on Agents, Founded in Logic

2004-02 Lai Xu (UvT)
Monitoring Multi-party Contracts for E-business

2004-03 Perry Groot (VU)
A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving

2004-04 Chris van Aart (UVA)
Organizational Principles for Multi-Agent Architectures

2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity

2004-06 Bart-Jan Hommes (TUD)
The Evaluation of Business Process Modeling Techniques

2004-07 Elise Boltjes (UM)
Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar
abstract denken, vooral voor meisjes

2004-08 Joop Verbeek(UM)
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale
politiële gegevensuitwisseling en digitale expertise

2004-09 Martin Caminada (VU)
For the Sake of the Argument; explorations into argument-based reasoning

2004-10 Suzanne Kabel (UVA)
Knowledge-rich indexing of learning-objects

2004-11 Michel Klein (VU)
Change Management for Distributed Ontologies

2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents

2004-13 Wojciech Jamroga (UT)
Using Multiple Models of Reality: On Agents who Know how to Play

2004-14 Paul Harrenstein (UU)
Logic in Conflict. Logical Explorations in Strategic Equilibrium

2004-15 Arno Knobbe (UU)
Multi-Relational Data Mining

2004-16 Federico Divina (VU)
Hybrid Genetic Relational Search for Inductive Learning

2004-17 Mark Winands (UM)
Informed Search in Complex Games

2004-18 Vania Bessa Machado (UvA)
Supporting the Construction of Qualitative Knowledge Models

2004-19 Thijs Westerveld (UT)

Using generative probabilistic models for multimedia retrieval

2004-20 Madelon Evers (Nyenrode)
Learning from Design: facilitating multidisciplinary design teams

2005

2005-01 Floor Verdenius (UVA)
Methodological Aspects of Designing Induction-Based Applications

2005-02 Erik van der Werf (UM))
AI techniques for the game of Go

2005-03 Franc Grootjen (RUN)
A Pragmatic Approach to the Conceptualisation of Language

2005-04 Nirvana Meratnia (UT)
Towards Database Support for Moving Object data

2005-05 Gabriel Infante-Lopez (UVA)
Two-Level Probabilistic Grammars for Natural Language Parsing

2005-06 Pieter Spronck (UM)
Adaptive Game AI

2005-07 Flavius Frasincar (TUE)
Hypermedia Presentation Generation for Semantic Web Information Systems

2005-08 Richard Vdovjak (TUE)
A Model-driven Approach for Building Distributed Ontology-based Web Applications

2005-09 Jeen Broekstra (VU)
Storage, Querying and Inferencing for Semantic Web Languages

2005-10 Anders Bouwer (UVA)
Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments

2005-11 Elth Ogston (VU)
Agent Based Matchmaking and Clustering - A Decentralized Approach to Search

2005-12 Csaba Boer (EUR)
Distributed Simulation in Industry

2005-13 Fred Hamburg (UL)
Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen

2005-14 Borys Omelayenko (VU)
Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics

2005-15 Tibor Bosse (VU)
Analysis of the Dynamics of Cognitive Processes

2005-16 Joris Graaumans (UU)
Usability of XML Query Languages

2005-17 Boris Shishkov (TUD)
Software Specification Based on Re-usable Business Components

2005-18 Danielle Sent (UU)
Test-selection strategies for probabilistic networks

2005-19 Michel van Dartel (UM)
Situated Representation

2005-20 Cristina Coteanu (UL)
Cyber Consumer Law, State of the Art and Perspectives

2005-21 Wijnand Derks (UT)
Improving Concurrency and Recovery in Database Systems by
Exploiting Application Semantics

2006

2006-01 Samuil Angelov (TUE)
Foundations of B2B Electronic Contracting

2006-02 Cristina Chisalita (VU)
Contextual issues in the design and use of information technology in organizations

2006-03 Noor Christoph (UVA)
The role of metacognitive skills in learning to solve problems

2006-04 Marta Sabou (VU)
Building Web Service Ontologies

2006-05 Cees Pierik (UU)
Validation Techniques for Object-Oriented Proof Outlines

2006-06 Ziv Baida (VU)
Software-aided Service Bundling - Intelligent Methods & Tools
for Graphical Service Modeling

2006-07 Marko Smiljanic (UT)
XML schema matching – balancing efficiency and effectiveness by means of clustering

2006-08 Eelco Herder (UT)
Forward, Back and Home Again - Analyzing User Behavior on the Web

2006-09 Mohamed Wahdan (UM)
Automatic Formulation of the Auditor's Opinion

2006-10 Ronny Siebes (VU)
Semantic Routing in Peer-to-Peer Systems

2006-11 Joeri van Ruth (UT)
Flattening Queries over Nested Data Types

2006-12 Bert Bongers (VU)
Interactivation - Towards an e-cology of people, our technological environment, and the arts

2006-13 Henk-Jan Lebbink (UU)
Dialogue and Decision Games for Information Exchanging Agents

2006-14 Johan Hoorn (VU)
Software Requirements: Update, Upgrade, Redesign -
towards a Theory of Requirements Change

2006-15 Rainer Malik (UU)
CONAN: Text Mining in the Biomedical Domain

2006-16 Carsten Riggelsen (UU)

Approximation Methods for Efficient Learning of Bayesian Networks

2006-17 Stacey Nagata (UU)
User Assistance for Multitasking with Interruptions on a Mobile Device

2006-18 Valentin Zhizhkun (UVA)
Graph transformation for Natural Language Processing

2006-19 Birna van Riemsdijk (UU)
Cognitive Agent Programming: A Semantic Approach

2006-20 Marina Velikova (UvT)
Monotone models for prediction in data mining

2006-21 Bas van Gils (RUN)
Aptness on the Web

2006-22 Paul de Vrieze (RUN)
Fundaments of Adaptive Personalisation

2006-23 Ion Juvina (UU)
Development of Cognitive Model for Navigating on the Web

2006-24 Laura Hollink (VU)
Semantic Annotation for Retrieval of Visual Resources

2006-25 Madalina Drugan (UU)
Conditional log-likelihood MDL and Evolutionary MCMC

2006-26 Vojkan Mihajlovic (UT)
Score Region Algebra: A Flexible Framework for Structured Information Retrieval

2006-27 Stefano Bocconi (CWI)
Vox Populi: generating video documentaries from semantically annotated media repositories

2006-28 Borkur Sigurbjornsson (UVA)
Focused Information Access using XML Element Retrieval

2007
2007-01 Kees Leune (UvT)
Access Control and Service-Oriented Architectures

2007-02 Wouter Teepe (RUG)
Reconciling Information Exchange and Confidentiality: A Formal Approach

2007-03 Peter Mika (VU)
Social Networks and the Semantic Web

2007-04 Jurriaan van Diggelen (UU)
Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach

2007-05 Bart Schermer (UL)
Software Agents, Surveillance, and the Right to Privacy:
a Legislative Framework for Agent-enabled Surveillance

2007-06 Gilad Mishne (UVA)
Applied Text Analytics for Blogs

2007-07 Natasa Jovanovic' (UT)
To Whom It May Concern - Addressee Identification in Face-to-Face Meetings

2007-08 Mark Hoogendoorn (VU)
Modeling of Change in Multi-Agent Organizations

2007-09 David Mobach (VU)
Agent-Based Mediated Service Negotiation

2007-10 Huib Aldewereld (UU)
Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols

2007-11 Natalia Stash (TUE)
Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System

2007-12 Marcel van Gerven (RUN)
Bayesian Networks for Clinical Decision Support:
A Rational Approach to Dynamic Decision-Making under Uncertainty

2007-13 Rutger Rienks (UT)
Meetings in Smart Environments; Implications of Progressing Technology

2007-14 Niek Bergboer (UM)
Context-Based Image Analysis

2007-15 Joyca Lacroix (UM)
NIM: a Situated Computational Memory Model

2007-16 Davide Grossi (UU)
Designing Invisible Handcuffs.
Formal investigations in Institutions and Organizations for Multi-agent Systems

2007-17 Theodore Charitos (UU)
Reasoning with Dynamic Networks in Practice

2007-18 Bart Orriens (UvT)
On the development an management of adaptive business collaborations

2007-19 David Levy (UM)
Intimate relationships with artificial partners

2007-20 Slinger Jansen (UU)
Customer Configuration Updating in a Software Supply Network

2007-21 Karianne Vermaas (UU)
Fast diffusion and broadening use:
A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005

2007-22 Zlatko Zlatev (UT)
Goal-oriented design of value and process models from patterns

2007-23 Peter Barna (TUE)
Specification of Application Logic in Web Information Systems

2007-24 Georgina Ramrez Camps (CWI)
Structural Features in XML Retrieval

2007-25 Joost Schalken (VU)
Empirical Investigations in Software Process Improvement

2008

2008-01 Katalin Boer-Sorbn (EUR)

Agent-Based Simulation of Financial Markets: A modular,continuous-time approach

2008-02 Alexei Sharpanskykh (VU)
On Computer-Aided Methods for Modeling and Analysis of Organizations

2008-03 Vera Hollink (UVA)
Optimizing hierarchical menus: a usage-based approach

2008-04 Ander de Keijzer (UT)
Management of Uncertain Data - towards unattended integration

2008-05 Bela Mutschler (UT)
Modeling and simulating causal dependencies on
process-aware information systems from a cost perspective

2008-06 Arjen Hommersom (RUN)
On the Application of Formal Methods to Clinical Guidelines,
an Artificial Intelligence Perspective

2008-07 Peter van Rosmalen (OU)
Supporting the tutor in the design and support of adaptive e-learning

2008-08 Janneke Bolt (UU)
Bayesian Networks: Aspects of Approximate Inference

2008-09 Christof van Nimwegen (UU)
The paradox of the guided user: assistance can be counter-effective

2008-10 Wauter Bosma (UT)
Discourse oriented summarization

2008-11 Vera Kartseva (VU)
Designing Controls for Network Organizations: A Value-Based Approach

2008-12 Jozsef Farkas (RUN)
A Semiotically Oriented Cognitive Model of Knowledge Representation

2008-13 Caterina Carraciolo (UVA)
Topic Driven Access to Scientific Handbooks

2008-14 Arthur van Bunningen (UT)
Context-Aware Querying; Better Answers with Less Effort

2008-15 Martijn van Otterlo (UT)
The Logic of Adaptive Behavior: Knowledge Representation and Algorithms
for the Markov Decision Process Framework in First-Order Domains.

2008-16 Henriette van Vugt (VU)
Embodied agents from a user's perspective

2008-17 Martin Op 't Land (TUD)
Applying Architecture and Ontology to the Splitting and Allying of Enterprises

2008-18 Guido de Croon (UM)
Adaptive Active Vision

2008-19 Henning Rode (UT)
From Document to Entity Retrieval: Improving Precision and Performance
of Focused Text Search

2008-20 Rex Arendsen (UVA)

Geen bericht, goed bericht.
Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven

2008-21 Krisztian Balog (UVA)
People Search in the Enterprise

2008-22 Henk Koning (UU)
Communication of IT-Architecture

2008-23 Stefan Visscher (UU)
Bayesian network models for the management of ventilator-associated pneumonia

2008-24 Zharko Aleksovski (VU)
Using background knowledge in ontology matching

2008-25 Geert Jonker (UU)
Efficient and Equitable Exchange in Air Traffic Management Plan Repair
using Spender-signed Currency

2008-26 Marijn Huijbregts (UT)
Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled

2008-27 Hubert Vogten (OU)
Design and Implementation Strategies for IMS Learning Design

2008-28 Ildiko Flesch (RUN)
On the Use of Independence Relations in Bayesian Networks

2008-29 Dennis Reidsma (UT)
Annotations and Subjective Machines - Of Annotators, Embodied Agents, Users, and Other Humans

2008-30 Wouter van Atteveldt (VU)
Semantic Network Analysis: Techniques for Extracting,
Representing and Querying Media Content

2008-31 Loes Braun (UM)
Pro-Active Medical Information Retrieval

2008-32 Trung H. Bui (UT)
Toward Affective Dialogue Management
using Partially Observable Markov Decision Processes

2008-33 Frank Terpstra (UVA)
Scientific Workflow Design; theoretical and practical issues

2008-34 Jeroen de Knijf (UU)
Studies in Frequent Tree Mining

2008-35 Ben Torben Nielsen (UvT)
Dendritic morphologies: function shapes structure

2009

2009-01 Rasa Jurgelenaite (RUN)
Symmetric Causal Independence Models

2009-02 Willem Robert van Hage (VU)
Evaluating Ontology-Alignment Techniques

2009-22 Pavel Serdyukov (UT)
Search For Expertise: Going beyond direct evidence

2009-23 Peter Hofgesang (VU)
Modelling Web Usage in a Changing Environment

2009-24 Annerieke Heuvelink (VUA)
Cognitive Models for Training Simulations

2009-25 Alex van Ballegooij (CWI)
"RAM: Array Database Management through Relational Mapping"

2009-26 Fernando Koch (UU)
An Agent-Based Model for the Development of Intelligent Mobile Services

2009-27 Christian Glahn (OU)
Contextual Support of social Engagement and Reflection on the Web

2009-28 Sander Evers (UT)
Sensor Data Management with Probabilistic Models

2009-29 Stanislav Pokraev (UT)
Model-Driven Semantic Integration of Service-Oriented Applications

2009-30 Marcin Zukowski (CWI)
Balancing vectorized query execution with bandwidth-optimized storage

2009-31 Sofiya Katrenko (UVA)
A Closer Look at Learning Relations from Text

2009-32 Rik Farenhorst (VU) and Remco de Boer (VU)
Architectural Knowledge Management: Supporting Architects and Auditors

2009-33 Khiet Truong (UT)
How Does Real Affect Affect Affect Recognition In Speech?

2009-34 Inge van de Weerd (UU)
Advancing in Software Product Management:
An Incremental Method Engineering Approach

2009-35 Wouter Koelewijn (UL)
Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling

2009-36 Marco Kalz (OUN)
Placement Support for Learners in Learning Networks

2009-37 Hendrik Drachsler (OUN)
Navigation Support for Learners in Informal Learning Networks

2009-38 Riina Vuorikari (OU)
Tags and self-organisation:
a metadata ecology for learning resources in a multilingual context

2009-39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin)
Service Substitution – A Behavioral Approach Based on Petri Nets

2009-40 Stephan Raaijmakers (UvT)
Multinomial Language Learning: Investigations into the Geometry of Language

2009-41 Igor Berezhnyy (UvT)

Digital Analysis of Paintings

2009-42 Toine Bogers
Recommender Systems for Social Bookmarking

2009-43 Virginia Nunes Leal Franqueira (UT)
Finding Multi-step Attacks in Computer Networks
using Heuristic Search and Mobile Ambients

2009-44 Roberto Santana Tapia (UT)
Assessing Business-IT Alignment in Networked Organizations

2009-45 Jilles Vreeken (UU)
Making Pattern Mining Useful

2009-46 Loredana Afanasiev (UvA)
Querying XML: Benchmarks and Recursion

2010

2010-01 Matthijs van Leeuwen (UU)
Patterns that Matter

2010-02 Ingo Wassink (UT)
Work flows in Life Science

2010-03 Joost Geurts (CWI)
A Document Engineering Model and Processing Framework for Multimedia documents

2010-04 Olga Kulyk (UT)
Do You Know What I Know?
Situational Awareness of Co-located Teams in Multidisplay Environments

2010-05 Claudia Hauff (UT)
Predicting the Effectiveness of Queries and Retrieval Systems

2010-06 Sander Bakkes (UvT)
Rapid Adaptation of Video Game AI

2010-07 Wim Fikkert (UT)
Gesture interaction at a Distance

2010-08 Krzysztof Siewicz (UL)
Towards an Improved Regulatory Framework of Free Software.
Protecting user freedoms in a world of software communities and eGovernments

2010-09 Hugo Kielman (UL)
A Politiele gegevensverwerking en Privacy, Naar een effectieve waarborging

2010-10 Rebecca Ong (UL)
Mobile Communication and Protection of Children

2010-11 Adriaan Ter Mors (TUD)
The world according to MARP: Multi-Agent Route Planning

2010-12 Susan van den Braak (UU)
Sensemaking software for crime analysis

2010-13 Gianluigi Folino (RUN)
High Performance Data Mining using Bio-inspired techniques

2010-14 Sander van Splunter (VU)
Automated Web Service Reconfiguration

2010-15 Lianne Bodenstaff (UT)
Managing Dependency Relations in Inter-Organizational Models

2010-16 Sicco Verwer (TUD)
Efficient Identification of Timed Automata, theory and practice

2010-17 Spyros Kotoulas (VU)
Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications

2010-18 Charlotte Gerritsen (VU)
Caught in the Act: Investigating Crime by Agent-Based Simulation

2010-19 Henriette Cramer (UvA)
People's Responses to Autonomous and Adaptive Systems

2010-20 Ivo Swartjes (UT)
Whose Story Is It Anyway?
How Improv Informs Agency and Authorship of Emergent Narrative

2010-21 Harold van Heerde (UT)
Privacy-aware data management by means of data degradation

2010-22 Michiel Hildebrand (CWI)
End-user Support for Access to Heterogeneous Linked Data

2010-23 Bas Steunebrink (UU)
The Logical Structure of Emotions

2010-24 Dmytro Tykhonov
Designing Generic and Efficient Negotiation Strategies

2010-25 Zulfiqar Ali Memon (VU)
Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective

2010-26 Ying Zhang (CWI)
XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines

2010-27 Marten Voulon (UL)
Automatisch contracteren

2010-28 Arne Koopman (UU)
Characteristic Relational Patterns

2010-29 Stratos Idreos(CWI)
Database Cracking: Towards Auto-tuning Database Kernels

2010-30 Marieke van Erp (UvT)
Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval

2010-31 Victor de Boer (UVA)
Ontology Enrichment from Heterogeneous Sources on the Web

2010-32 Marcel Hiel (UvT)
An Adaptive Service Oriented Architecture:
Automatically solving Interoperability Problems

2010-33 Robin Aly (UT)
Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval

2010-34 Teduh Dirgahayu (UT)
Interaction Design in Service Compositions

2010-35 Dolf Trieschnigg (UT)
Proof of Concept: Concept-based Biomedical Information Retrieval

2010-36 Jose Janssen (OU)
Paving the Way for Lifelong Learning;
Facilitating competence development through a learning path specification

2010-37 Niels Lohmann (TUE)
Correctness of services and their composition

2010-38 Dirk Fahland (TUE)
From Scenarios to components

2010-39 Ghazanfar Farooq Siddiqui (VU)
Integrative modeling of emotions in virtual agents

2010-40 Mark van Assem (VU)
Converting and Integrating Vocabularies for the Semantic Web

2010-41 Guillaume Chaslot (UM)
Monte-Carlo Tree Search

2010-42 Sybren de Kinderen (VU)
Needs-driven service bundling in a multi-supplier setting -
the computational e3-service approach

2010-43 Peter van Kranenburg (UU)
A Computational Approach to Content-Based Retrieval of Folk Song Melodies

2010-44 Pieter Bellekens (TUE)
An Approach towards Context-sensitive and User-adapted Access
to Heterogeneous Data Sources, Illustrated in the Television Domain

2010-45 Vasilios Andrikopoulos (UvT)
A theory and model for the evolution of software services

2010-46 Vincent Pijpers (VU)
e3alignment: Exploring Inter-Organizational Business-ICT Alignment

2010-47 Chen Li (UT)
Mining Process Model Variants: Challenges, Techniques, Examples

2010-48 Milan Lovric (EUR)
Behavioral Finance and Agent-Based Artificial Markets

2010-49 Jahn-Takeshi Saito (UM)
Solving difficult game positions

2010-50 Bouke Huurnink (UVA)
Search in Audiovisual Broadcast Archives

2010-51 Alia Khairia Amin (CWI)
Understanding and supporting information seeking tasks in multiple sources

2010-52 Peter-Paul van Maanen (VU)

Adaptive Support for Human-Computer Teams:
Exploring the Use of Cognitive Models of Trust and Attention

2010-53 Edgar Meij (UVA)
Combining Concepts and Language Models for Information Access

2011

2011-01 Botond Cseke (RUN)
Variational Algorithms for Bayesian Inference in Latent Gaussian Models

2011-02 Nick Tinnemeier(UU)
Organizing Agent Organizations.
Syntax and Operational Semantics of an Organization-Oriented Programming Language

2011-03
Jan Martijn van der Werf (TUE)
Compositional Design and Verification of Component-Based Information Systems

2011-04 Hado van Hasselt (UU)
Insights in Reinforcement Learning;
Formal analysis and empirical evaluation of temporal-difference learning algorithms

2011-05 Base van der Raadt (VU)
Enterprise Architecture Coming of Age -
Increasing the Performance of an Emerging Discipline.

2011-06 Yiwen Wang (TUE)
Semantically-Enhanced Recommendations in Cultural Heritage

2011-07 Yujia Cao (UT)
Multimodal Information Presentation for High Load Human Computer Interaction

2011-08 Nieske Vergunst (UU)
BDI-based Generation of Robust Task-Oriented Dialogues 2011-09 Tim de Jong (OU)
Contextualised Mobile Media for Learning

2011-10 Bart Bogaert (UvT)
Cloud Content Contention

2011-11 Dhaval Vyas (UT)
Designing for Awareness: An Experience-focused HCI Perspective

2011-12 Carmen Bratosin (TUE)
Grid Architecture for Distributed Process Mining

2011-13 Xiaoyu Mao (UvT)
Airport under Control. Multiagent Scheduling for Airport Ground Handling

2011-14 Milan Lovric (EUR)
Behavioral Finance and Agent-Based Artificial Markets

2011-15 Marijn Koolen (UvA)
The Meaning of Structure: the Value of Link Evidence for Information Retrieval

2011-16 Maarten Schadd (UM)
Selective Search in Games of Different Complexity

2011-17 Jiyin He (UVA)
Exploring Topic Structure: Coherence, Diversity and Relatedness

2011-18 Mark Ponsen (UM)
Strategic Decision-Making in complex games

2011-19 Ellen Rusman (OU)
The Mind ' s Eye on Personal Profiles

2011-20 Qing Gu (VU)
Guiding service-oriented software engineering - A view-based approach

2011-21 Linda Terlouw (TUD)
Modularization and Specification of Service-Oriented Systems

2011-22 Junte Zhang (UVA)
System Evaluation of Archival Description and Access

2011-23 Wouter Weerkamp (UVA)
Finding People and their Utterances in Social Media

2011-24 Herwin van Welbergen (UT)
Behavior Generation for Interpersonal Coordination
with Virtual Humans On Specifying, Scheduling
and Realizing Multimodal Virtual Human Behavior

2011-25 Syed Waqar ul Qounain Jaffry (VU)
Analysis and Validation of Models for Trust Dynamics

2011-26 Matthijs Aart Pontier (VU)
Virtual Agents for Human Communication -
Emotion Regulation and Involvement-Distance Trade-Offs
in Embodied Conversational Agents and Robots

2011-27 Aniel Bhulai (VU)
Dynamic website optimization through autonomous management of design patterns

2011-28 Rianne Kaptein(UVA)
Effective Focused Retrieval by Exploiting Query Context and Document Structure

2011-29 Faisal Kamiran (TUE)
Discrimination-aware Classification

2011-30 Egon van den Broek (UT)
Affective Signal Processing (ASP): Unraveling the mystery of emotions

2011-31 Ludo Waltman (EUR)
Computational and Game-Theoretic Approaches for Modeling Bounded Rationality

2011-32 Nees-Jan van Eck (EUR)
Methodological Advances in Bibliometric Mapping of Science

2011-33 Tom van der Weide (UU)
Arguing to Motivate Decisions

2011-34 Paolo Turrini (UU)
Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations

2011-35 Maaike Harbers (UU)
Explaining Agent Behavior in Virtual Training

2011-36 Erik van der Spek (UU)
Experiments in serious game design: a cognitive approach

2011-37 Adriana Burlutiu (RUN)
Machine Learning for Pairwise Data,
Applications for Preference Learning and Supervised Network Inference

2011-38 Nyree Lemmens (UM)
Bee-inspired Distributed Optimization

2011-39 Joost Westra (UU)
Organizing Adaptation using Agents in Serious Games

2011-40 Viktor Clerc (VU)
Architectural Knowledge Management in Global Software Development

2011-41 Luan Ibraimi (UT)
Cryptographically Enforced Distributed Data Access Control

2011-42 Michal Sindlar (UU)
Explaining Behavior through Mental State Attribution

2011-43 Henk van der Schuur (UU)
Process Improvement through Software Operation Knowledge

2011-44 Boris Reuderink (UT)
Robust Brain-Computer Interfaces

2011-45 Herman Stehouwer (UvT)
Statistical Language Models for Alternative Sequence Selection

2011-46 Beibei Hu (TUD)
Towards Contextualized Information Delivery:
A Rule-based Architecture for the Domain of Mobile Police Work

2011-47 Azizi Bin Ab Aziz(VU)
Exploring Computational Models for Intelligent Support of Persons with Depression

2011-48 Mark Ter Maat (UT)
Response Selection and Turn-taking for a Sensitive Artificial Listening Agent

2011-49 Andreea Niculescu (UT)
Conversational interfaces for task-oriented spoken dialogues:
design aspects influencing interaction quality

2012

2012-01 Terry Kakeeto (UvT)
Relationship Marketing for SMEs in Uganda

2012-02 Muhammad Umair(VU)
Adaptivity, emotion, and Rationality in Human and Ambient Agent Models

2012-03 Adam Vanya (VU)
Supporting Architecture Evolution by Mining Software Repositories

2012-04 Jurriaan Souer (UU)
Development of Content Management System-based Web Applications

2012-05 Marijn Plomp (UU)
Maturing Interorganisational Information Systems

2012-06 Wolfgang Reinhardt (OU)

Awareness Support for Knowledge Workers in Research Networks

2012-07 Rianne van Lambalgen (VU)
When the Going Gets Tough:
Exploring Agent-based Models of Human Performance under Demanding Conditions

2012-08 Gerben de Vries (UVA)
Kernel Methods for Vessel Trajectories

2012-09 Ricardo Neisse (UT)
Trust and Privacy Management Support for Context-Aware Service Platforms

2012-10 David Smits (TUE)
Towards a Generic Distributed Adaptive Hypermedia Environment

2012-11 J.C.B. Rantham Prabhakara (TUE)
Process Mining in the Large: Preprocessing, Discovery, and Diagnostics

2012-12 Kees van der Sluijs (TUE)
Model Driven Design and Data Integration in Semantic Web Information Systems

2012-13 Suleman Shahid (UvT)
Fun and Face: Exploring non-verbal expressions of emotion during playful interactions

2012-14 Evgeny Knutov(TUE)
Generic Adaptation Framework for Unifying Adaptive Web-based Systems

2012-15 Natalie van der Wal (VU)
Social Agents. Agent-Based Modelling of
Integrated Internal and Social Dynamics of Cognitive and Affective Processes

2012-16 Fiemke Both (VU)
Helping people by understanding them -
Ambient Agents supporting task execution and depression treatment

2012-17 Amal Elgammal (UvT)
Towards a Comprehensive Framework for Business Process Compliance

2012-18 Eltjo Poort (VU)
Improving Solution Architecting Practices

2012-19 Helen Schonenberg (TUE)
What's Next? Operational Support for Business Process Execution

2012-20 Ali Bahramisharif (RUN)
Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing

2012-21 Roberto Cornacchia (TUD)
Querying Sparse Matrices for Information Retrieval

2012-22 Thijs Vis (UvT)
Intelligence, politie en veiligheidsdienst: verenigbare grootheden?

2012-23 Christian Muehl (UT)
Toward Affective Brain-Computer Interfaces:
Exploring the Neurophysiology of Affect during Human Media Interaction

2012-24 Laurens van der Werff (UT)
Evaluation of Noisy Transcripts for Spoken Document Retrieval

2012-25 Silja Eckartz (UT)

2012-45 Benedikt Kratz (UvT)
A Model and Language for Business-aware Transactions

2012-46 Simon Carter (UVA)
Exploration and Exploitation of Multilingual Data for Statistical Machine Translation

2012-47 Manos Tsagkias (UVA)
Mining Social Media: Tracking Content and Predicting Behavior

2012-48 Jorn Bakker (TUE)
Handling Abrupt Changes in Evolving Time-series Data

2012-49 Michael Kaisers (UM)
Learning against Learning -
Evolutionary dynamics of reinforcement learning algorithms in strategic interactions

2012-50 Steven van Kervel (TUD)
Ontologogy driven Enterprise Information Systems Engineering

2012-51 Jeroen de Jong (TUD)
Heuristics in Dynamic Scheduling;
a practical framework with a case study in elevator dispatching

2013

2013-01 Viorel Milea (EUR)
News Analytics for Financial Decision Support

2013-02 Erietta Liarou (CWI)
MonetDB/DataCell:
Leveraging the Column-store Database Technology
for Efficient and Scalable Stream Processing

2013-03 Szymon Klarman (VU)
Reasoning with Contexts in Description Logics

2013-04 Chetan Yadati(TUD)
Coordinating autonomous planning and scheduling

2013-05 Dulce Pumareja (UT)
Groupware Requirements Evolutions Patterns

2013-06 Romulo Goncalves(CWI)
The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience

2013-07 Giel van Lankveld (UvT)
Quantifying Individual Player Differences

2013-08 Robbert-Jan Merk(VU)
Making enemies: cognitive modeling for opponent agents in fighter pilot simulators

2013-09 Fabio Gori (RUN)
Metagenomic Data Analysis: Computational Methods and Applications

2013-10 Jeewanie Jayasinghe Arachchige(UvT)
A Unified Modeling Framework for Service Design.

2013-11 Evangelos Pournaras(TUD)
Multi-level Reconfigurable Self-organization in Overlay Services

2013-12 Marian Razavian(VU)

Knowledge-driven Migration to Services

2013-13 Mohammad Safiri(UT)
Service Tailoring: User-centric creation of integrated IT-based homecare services
to support independent living of elderly

2013-14 Jafar Tanha (UVA)
Ensemble Approaches to Semi-Supervised Learning Learning

2013-15 Daniel Hennes (UM)
Multiagent Learning - Dynamic Games and Applications

2013-16 Eric Kok (UU)
Exploring the practical benefits of argumentation in multi-agent deliberation

2013-17 Koen Kok (VU)
The PowerMatcher: Smart Coordination for the Smart Electricity Grid

2013-18 Jeroen Janssens (UvT)
Outlier Selection and One-Class Classification

2013-19 Renze Steenhuizen (TUD)
Coordinated Multi-Agent Planning and Scheduling

2013-20 Katja Hofmann (UvA)
Fast and Reliable Online Learning to Rank for Information Retrieval

2013-21 Sander Wubben (UvT)
Text-to-text generation by monolingual machine translation

2013-22 Tom Claassen (RUN)
Causal Discovery and Logic

2013-23 Patricio de Alencar Silva(UvT)
Value Activity Monitoring

2013-24 Haitham Bou Ammar (UM)
Automated Transfer in Reinforcement Learning

2013-25 Agnieszka Anna Latoszek-Berendsen (UM)
Intention-based Decision Support.
A new way of representing and implementing clinical guidelines
in a Decision Support System

2013-26 Alireza Zarghami (UT)
Architectural Support for Dynamic Homecare Service Provisioning

2013-27 Mohammad Huq (UT)
Inference-based Framework Managing Data Provenance

2013-28 Frans van der Sluis (UT)
When Complexity becomes Interesting: An Inquiry into the Information eXperience

2013-29 Iwan de Kok (UT)
Listening Heads

2013-30 Joyce Nakatumba (TUE)
Resource-Aware Business Process Management: Analysis and Support

2013-31 Dinh Khoa Nguyen (UvT)
Blueprint Model and Language for Engineering Cloud Applications

2013-32 Kamakshi Rajagopal (OUN)
Networking For Learning;
The role of Networking in a Lifelong Learner's Professional Development

2013-33 Qi Gao (TUD)
User Modeling and Personalization in the Microblogging Sphere

2013-34 Kien Tjin-Kam-Jet (UT)
Distributed Deep Web Search

2013-35 Abdallah El Ali (UvA)
Minimal Mobile Human Computer Interaction

2013-36 Than Lam Hoang (TUe)
Pattern Mining in Data Streams

2013-37 Dirk Börner (OUN)
Ambient Learning Displays

2013-38 Eelco den Heijer (VU)
Autonomous Evolutionary Art

2013-39 Joop de Jong (TUD)
A Method for Enterprise Ontology based Design of Enterprise Information Systems

2013-40 Pim Nijssen (UM)
Monte-Carlo Tree Search for Multi-Player Games

2013-41 Jochem Liem (UVA)
Supporting the Conceptual Modelling of Dynamic Systems:
A Knowledge Engineering Perspective on Qualitative Reasoning

2013-42 Leon Planken (TUD)
Algorithms for Simple Temporal Reasoning

2013-43 Marc Bron (UVA)
Exploration and Contextualization through Interaction and Concepts

2014

2014-01 Nicola Barile (UU)
Studies in Learning Monotone Models from Data

2014-02 Fiona Tuliyano (RUN)
Combining System Dynamics with a Domain Modeling Method

2014-03 Sergio Raul Duarte Torres (UT)
Information Retrieval for Children: Search Behavior and Solutions

2014-04 Hanna Jochmann-Mannak (UT)
Websites for children: search strategies and interface design -
Three studies on children's search performance and evaluation

2014-05 Jurriaan van Reijsen (UU)
Knowledge Perspectives on Advancing Dynamic Capability

2014-06 Damian Tamburri (VU)
Supporting Networked Software Development

2014-07 Arya Adriansyah (TUE)
Aligning Observed and Modeled Behavior

2014-08 Samur Araujo (TUD)
Data Integration over Distributed and Heterogeneous Data Endpoints

2014-09 Philip Jackson (UvT)
Toward Human-Level Artificial Intelligence:
Representation and Computation of Meaning in Natural Language

2014-10 Ivan Salvador Razo Zapata (VU)
Service Value Networks

2014-11 Janneke van der Zwaan (TUD)
An Empathic Virtual Buddy for Social Support

2014-12 Willem van Willigen (VU)
Look Ma, No Hands: Aspects of Autonomous Vehicle Control

2014-13 Arlette van Wissen (VU)
Agent-Based Support for Behavior Change:
Models and Applications in Health and Safety Domains

2014-14 Yangyang Shi (TUD)
Language Models With Meta-information

2014-15 Natalya Mogles (VU)
Agent-Based Analysis and Support of Human Functioning
in Complex Socio-Technical Systems: Applications in Safety and Healthcare

2014-16 Krystyna Milian (VU)
Supporting trial recruitment and design by automatically interpreting eligibility criteria

2014-17 Kathrin Dentler (VU)
Computing healthcare quality indicators automatically:
Secondary Use of Patient Data and Semantic Interoperability

2014-18 Mattijs Ghijsen (VU)
Methods and Models for the Design and Study of Dynamic Agent Organizations

2014-19 Vinicius Ramos (TUE)
Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support

2014-20 Mena Habib (UT)
Named Entity Extraction and Disambiguation for Informal Text: The Missing Link

2014-21 Kassidy Clark (TUD)
Negotiation and Monitoring in Open Environments

2014-22 Marieke Peeters (UU)
Personalized Educational Games - Developing agent-supported scenario-based training

2014-23 Eleftherios Sidirourgos (UvA/CWI)
Space Efficient Indexes for the Big Data Era

2014-24 Davide Ceolin (VU)
Trusting Semi-structured Web Data

2014-25 Martijn Lappenschaar (RUN)
New network models for the analysis of disease interaction

2014-46 Ke Tao (TUD)
Social Web Data Analytics: Relevance, Redundancy, Diversity

2014-47 Shangsong Liang (UVA)
Fusion and Diversification in Information Retrieval

2015

2015-01 Niels Netten (UvA)
Machine Learning for Relevance of Information in Crisis Response

2015-02 Faiza Bukhsh (UvT)
Smart auditing: Innovative Compliance Checking in Customs Controls

2015-03 Twan van Laarhoven (RUN)
Machine learning for network data

2015-04 Howard Spoelstra (OUN)
Collaborations in Open Learning Environments

2015-05 Christoph Bsch(UT)
Cryptographically Enforced Search Pattern Hiding

2015-06 Farideh Heidari (TUD)
Business Process Quality Computation -
Computing Non-Functional Requirements to Improve Business Processes

2015-07 Maria-Hendrike Peetz(UvA)
Time-Aware Online Reputation Analysis

2015-08 Jie Jiang (TUD)
Organizational Compliance: An agent-based model
for designing and evaluating organizational interactions

2015-09 Randy Klaassen(UT)
HCI Perspectives on Behavior Change Support Systems

2015-10 Henry Hermans (OUN)
OpenU: design of an integrated system to support lifelong learning

2015-11 Yongming Luo(TUE)
Designing algorithms for big graph datasets: A study of computing bisimulation and joins

2015-12 Julie Birkholz (VU)
Modi Operandi of Social Network Dynamics:
The Effect of Context on Scientific Collaboration Networks