EnviroInfo 2011: Innovations in Sharing Environmental Observations and Information
Copyright 2011 Shaker Verlag Aachen, ISBN: 978-3-8440-0451-9

Measurement and Rating of Software Induced Energy Consumption of Desktop PCs and Servers

Markus Dick, Eva Kern, Jakob Drangmeister, Stefan Naumann, Timo Johann[1]

Abstract

Up to now, the power consumption of ICT is still increasing. However, it is not clear whether or not energy savings through ICT overcompensate this increasing energy consumption. There are manifold efforts of Green IT addressing energy efficiency of computer hardware, but there is a lack of models, descriptions, or realizations in the field of computer software. In our paper, we present a method to measure and rate software-induced energy consumption of stand-alone applications on desktop computers as well as interactive transaction-based applications on servers. In both cases, realistic workloads are applied. Our test rig contains the tested system, an appropriate power meter, a workload generator, and a data evaluation system. Our measurement method is intended to support software developers, purchasers, administrators, and users in making informed decisions on software architecture and implementation as well as on software products they use or plan to use. As a proof of concept, we will describe two exemplary measurements that show the influence of software use on power consumption.

1. Introduction

The Brundtland report talks about a lifestyle "that meets the needs of the present without compromising the ability of future generations to meet their own needs." (UN General Assembly 1987, 51). In almost every branch of industry, the awareness for the necessity of Sustainable Development (abbr. SD) grows to consolidate mankind's future (Strange/Bayley 2008).

In the area of Information and Communication Technology (abbr. ICT), environmental sustainability recently attains a lot of importance. The potential of ICTs is expressed in the concepts Green IT and Green by IT. Green IT means: being environmentally sustainable in ICT, whereas Green by IT means: being environmentally sustainable by using ICT in order to gain higher energy and resource savings in its application areas (e.g. smart heating, smart logistics, e-paper, teleconferencing) (Erdmann/Hilty/Goodman 2004, GeSI 2008, OECD 2010).

Even if previous academic research discussed the question whether energy savings by ICT exceed energy consumption by ICT or not, e.g. due to more efficient processes or simulations of scenarios (Göhring 2004, Coroama/Hilty 2009), no consensus has been reached yet. Today, Green IT efforts are mainly focusing on hardware, even though software plays a significant role regarding energy efficient use of hardware, or early hardware obsolescence due to higher hardware requirements of new software (Hilty/Köhler/von Schéele 2006).

In our paper, we propose a measurement and rating method for software induced energy consumption. Its objectives are to support software developers during software development, purchasers when evaluating tenders for software products, as well as administrators and users in making informed decisions on software products they currently operate and use, or plan to operate and use in the future.

Our contribution fits into the GREENSOFT Model (Dick/Naumann/Kuhn 2010), a conceptual reference model for "Green and Sustainable Software".

[1] Umwelt-Campus Birkenfeld, Trier University of Applied Sciences, Institute for Software Systems
Campusallee, D-55761 Birkenfeld, Germany, email: greensoft@umwelt-campus.de

2. Background

Today, many publications discuss the relationship between ICT and SD, particularly environmental sustainability. Some of these publications outline in which way the recognition of sustainability issues can be integrated into software development processes (Dick/Naumann 2010, Albertao/Xiao/Tian 2010). However, these frameworks lack practical measurement methods that support persons responsible for identifying energy efficiency problems or complex sustainability issues in their source code and software systems.

Nevertheless, there are tools available, e.g. “Green Tracker” that support users in making informed decisions on the software they use by measuring and estimating the energy consumption of programs installed on the user’s computer (Amsel/Tomlinson 2010). This does not only support people in their decisions, but also raises users as well as software developers awareness of energy efficiency issues.

Relating to these issues, it was shown that different software products implementing a similar functionality, significantly differ in energy efficiency (Capra/Formenti/Francalanci 2010). It was also shown that energy efficiency of software applications depends on the software stack (compilers, frameworks, operating systems) used to realize this functionality. In one example, an application that induces a lower energy consumption as its competitor on one operating system induced a higher energy consumption on other operating systems and vice versa.

ISO developed a method for measurement and rating of performance of computer-based systems (ISO/IEC 14756:1999). It describes the measurement experiment, a workload model, the validation of the generated workloads, as well as the rating of the measured performance data. Although the rating of measured performance data is out of the focus of the method described in this paper, some parts of the standard, especially the workload model and the validation of the generated workloads are reused.

3. Areas of Application and Requirements for a Measurement Method

The proposed measurement and rating method for software induced energy consumption has the objective to support

- software developers during software development,
- purchasers when evaluating tenders for software products, as well as
- administrators and users in making informed decisions on software products they operate and use.

First, the proposed method should be practicable during software development, in order to support software architects and developers to address and increase energy efficiency of software systems currently under development. Appropriate tools during development are early processing time estimations and energy consumption measurements. The former can be addressed by the Software Performance Engineering method (Smith/Williams 2002), the latter by the method described here. If software developers decide on architectures or implementation details during development, in order to increase energy efficiency, as it was proposed by the software development process extension for Green and Sustainable Software Engineering (Dick/Naumann 2010), it is necessary to proof whether or not the measures taken achieve or even exceed the expected energy efficiency gains. In order to find software components that should be optimized, our method may also be used. For this purpose however, other methods and tools, e.g. performance profiling or source code instrumentation, may be more appropriate (Tayeb/Bross/Chang 2010).

Second, the proposed method should support purchasers who evaluate tenders for software products in public as well as in private organizations. Usually, such procurements can be divided into the procurement of custom (or bespoke) software and the procurement of standard software. For custom software, energy efficiency issues can be defined as non-functional software requirements, as it is done with performance requirements. During the procurement procedure, a bidding organization can proof that it has the expertise to handle such requirements by showing that it applies an appropriate quality assurance process and appropriate methods during development. For standard software products, energy efficiency issues can be used either as technical requirements or as award criteria. In this case, it is necessary to provide bidders with a standardized measurement method, so that their measurement results are comparable and can therefore be used in the procurement process for evaluation purposes.

Third, the proposed method should support home users, administrators, and purchasers in organizations that do not apply sophisticated tendering procedures in making informed decisions on software products they use or operate, or plan to use or operate in the future. It is clear that applying a complex measurement method, as it is described here, is not an option for those stakeholders. Thus, the according information should be printed on product boxes or product sheets by manufacturers. In this way, the customers can make informed decisions on which product fits their needs best. In the future, there may be a product label that indicates that a software product meets certain energy efficiency requirements.

The intended application areas as described above lead to basic requirements that should be fulfilled by the measurement method:

1. It should be independent of source code availability and it should not rely on source code instrumentation, because the source code is usually not available to the public
2. It should use typical workloads that are expected to appear if the product is used in its intended area of application
3. The workloads and tests should be statistically reproducible
4. It should provide statistically significant evidence, whether the induced energy consumption of two compared software systems is equal for a specific workload or not.

In principle, the measurement method should be applicable for different types of software products, which are e.g. categorized by Sommerville (2011, 10). However, embedded software or applications for portable computers are currently not covered by our method, because naturally in such environments energy is a strongly limited resource, and so energy aware programming is common practice. Instead, the method described in this paper tries to increase the importance of energy efficiency issues for other types of software, which is not common today. Thus, this method focuses on stand-alone applications that run on local computers, e.g. desktop PCs as well as interactive, transaction-based applications that run remotely on servers and are accessed by users from their own PCs. With minor changes in toolsets, it should also be possible to use this method to assess energy efficiency of batch processing systems, entertainment systems, and systems for modeling and simulation.

4. Test Rig and Measurement Method

4.1 Overview

The measurement environment consists of several hardware and software components that are necessary to perform the test (see Figure 1).

The System Under Test (abbr. SUT) is the computer hosting the application program, whose induced energy consumption is to be evaluated. The SUT is defined by the combination of hardware components that make up the computer system, the operating system, the runtime environment, and the application program. The application program uses a specific program interpreter, if it is written in an interpreted programming language, or a specific compiler to create the binary executable program. Any additional services and frameworks besides the operating system necessary for program execution belong to the runtime environment.

The Workload Generator (abbr. WG) is a computer program that generates the statistically reproducible workload that is applied to the SUT. Depending on the type of application program, it is either run directly on the SUT, e.g. for stand-alone applications or on one or more separate computers, e.g. as clients that access an interactive, transaction-based application on a remote server.

The Power Meter (abbr. PM) is an appropriate energy or power meter, whose readings can be accessed remotely to be aggregated and evaluated by the Data Evaluator and Aggregator (abbr. DAE). The PM is only connected to the SUT. The energy consumption of clients that access a remote server is not considered, as they require their own measurement.

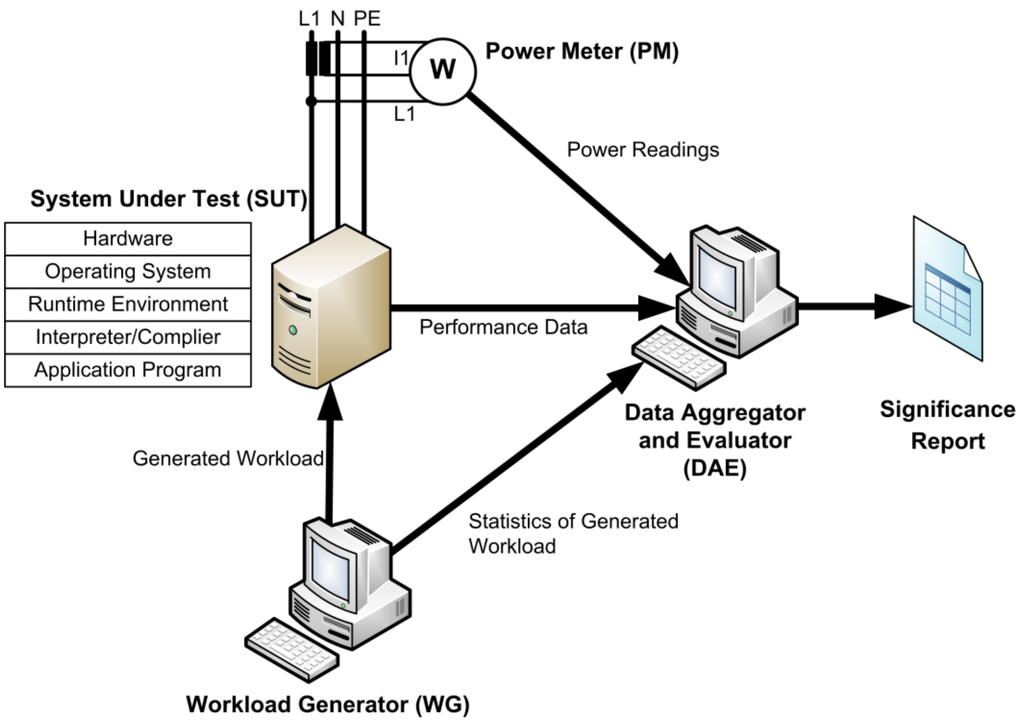

Figure 1
Overview of the measurement setup (SUT hierarchy adapted from Dirlewanger 2006, 109)

The DAE collects all data and evaluates it. This includes the power or energy readings, the performance readings from the SUT, and the workload execution statistics from the WG. In the case of two competing SUTs, a statistical significance report is generated that states, which SUT has the lower energy consumption and therefore is more energy efficient. If there is only one SUT, the measured data may be used for visualization in graphs, in order to support developers in finding software components that should be optimized. This option is not directly covered by this method.

4.2 Modeling and Defining Workloads

The workload model that is used by our method follows the workload model described by the ISO Standard (ISO/IEC 14756:1999). The basic idea of the model is that users execute several task chains (one could also call them workflows), which consist of several tasks, which themselves are defined by a specific activity performed by the user and the preparation time (one could also call it “think-time”). Due to the fact that we need to emulate users of different kinds, the workload model defines user types. For each user type, one can define different task preparation times. These preparation times are defined by mean and standard deviation. Each user can also execute several task chains. For each user type, the relative frequency of task chain types is defined. The sum of the relative chain frequencies for one user type must be one. A complete workload definition also includes the number of users and their type, which should be emulated by the WG. Additionally, a set of precision parameters define the maximum acceptable difference between the statistical parameters defined in the workload and the actual values achieved during workload emulation. These parameters are used to ensure the accuracy of the WG during validation.

In industry standard benchmarks that focus on system performance, synthetic workloads are usually used to test different areas of a software system. This may be sufficient from a general point of view, e.g. for testing performance under heavy loads, but may be insufficient from an organization's point of view, if e.g. energy efficiency is used in a tendering procedure. Here, it is necessary to consider the main purposes for which the application is intended to be used in an organization and to consider organization specific workloads that usually are expected to occur. Hence, realistic and typical workloads for a specific application area that are statistically reproducible should be used in order to assess energy efficiency.

4.3 Validating and Evaluating the Measurement Results

4.3.1 Validating the Measurement Results

The validation of the measurement ensures that the generated workload is statistically reproducible. The ISO method divides it into 3 different parts: prove correct work of the SUT, prove accuracy of the workloads generated by the WG, prove that the measurement interval is long enough to get reliable performance values (task execution mean times).

The first part is accomplished by checking the results computed by the SUT for each activity type against the expected results. Therefore, the expected results should be defined along with the workload description. The second part is accomplished by applying 3 checks: the maximum relative difference of the task chain frequencies, the maximum difference of the mean preparation times, and the maximum relative difference of the standard deviations of the preparation times. These checks are carried out for each combination of user type and chain type that is defined in the parameter set of a workload definition. For each check, the measured value is compared against the defined value.

The last part checks that the measurement interval is long enough in order to get a statistical significant mean execution time for each task type. The test checks if a given confidence interval covers the true but unknown mean execution time for each task type or not. Although the mean execution times are, as performance parameters, not of importance for comparing energy consumption of two SUTs, it can be used to decide whether or not the measurement interval is long enough and to decide if a workload is too heavy to be handled by an SUT (Dirlewanger 2006, 58). Hence, we propose that this check should be optional in the measurement process described here.

4.3.2 Evaluating the Measurement Results

In order to compare the mean energy consumption induced by two different SUTs, several measurement runs for each SUT are necessary and for each, the energy consumption has to be determined. Then it is determined, whether or not there is a statistical significant difference in mean energy consumption between the two systems. For this purpose, a two sided t-test can be applied on the mean of the sampled energy consumption in order to prove if the expected energy consumption of the two SUTs is equal or not (Field 2009). The hypotheses are defined as follows:

– Null Hypothesis: The mean energy consumption induced by SUT 1 and SUT 2 is equal
– Alternative Hypothesis: The mean energy consumption induced by SUT 1 and SUT 2 is not equal

The level of significance should be chosen from the common values (e.g. $\alpha=0.05$, $\alpha=0.01$).

4.4 Measuring Software on Servers

First, as a prerequisite for each measurement experiment, a clean setup of the SUT is necessary, so that no relicts of former experiments bias the current experiment. Concerning the workload, an adequate WG for servers must have the ability to execute several requests in parallel, in order to simulate multiple users who access the system. Additionally, a WG must generate log output that can be used to perform the necessary validations. The length of the measurement interval should be defined statically, e.g. 10 minutes. The appropriate length may be determined according to the third part of the validation procedure.

4.4.1 Experimental Setup

In order to prove our measurement method and experimental setup for server tests, we decided to measure web pages that are dynamically generated by content management systems. The basis of our server tests is a Supermicro server (P4BP8-G2) with two Intel Xeon dual core CPUs (@2.4 GHz, 2 GB RAM, 40 GB HDD). For the server tests, we decided to use an Ubuntu GNU/Linux (SMP 10.04 LTS, Kernel 2.6.32-32-generic-pae), because GNU/Linux based systems are widely offered by hosting providers.

An appropriate WG for testing web pages, is Apache JMeter (http://jakarta.apache.org/jmeter/)[2]. Its logging mechanism is capable of logging all information that is needed to validate the measurement results. Unfortunately, the pause times of its pause elements are not logged out of the box. Hence, it is necessary to implement this functionality via a custom plug-in.

The performance data of the SUT is collected by using the SYSSTAT[3] utilities. These include the tool “pidstat”, which can be used to report I/O statistics, memory, and CPU usage on a process level. Additionally, the tool “sar” is included, which reports the similar statistics on a system level. The performance data may be also used to determine the mean CPU utilization that is caused by a workload.

4.4.2 Exemplary Measurements and Results with Web CMSs

As an example measurement we decided to compare two configurations of the PHP-based web content management system *Joomla!* (1.5.23, http://www.joomla.org/). As prerequisites, it requires a web server (Apache httpd 2.2.14), a PHP runtime environment (PHP 5.3.2), and a relational database system (MySQL 5.1). One configuration uses a hard disk cache that stores HTML fragments of already accessed web pages, so that there is no need to query the information from the database and to convert it into HTML pages. The other configuration, the default configuration, does not use such a cache, so that the information is fetched from the database and processed for each request. Both configurations do not enable caching in web browsers by embedding meta data into the HTML output or into the HTTP header.

The website that was used for the experiment consisted of several legal documents of the European Commission (see Figure 2). These documents were arranged in two categories. The categories are presented as menu items in the menu tree of the website. The web pages Homepage, Directives, and References are also presented as menu items, but not the single legal documents. They are instead presented in tables as hyperlinks, which are dynamically generated by the CMS for each category on a dedicated web page.

The workload consists of one user type and one task chain. In the task chain, four web pages are accessed several times: the Directives page 2 times, the Climate Change page 2 times, the Renewable Energy page 4 times, and one of the legal documents two times. The workload starts 67[4] threads, which

[2] All product websites were last accessed at 2011-07-22

[3] http://sebastien.godard.pagesperso-orange.fr/

[4] This number was determined by experiment: with more threads the validation failed due to loss of accuracy in preparation times.

represent a realization of a user type, during a period of 2 minutes. Afterwards, the measurement interval begins. It ends after 10 minutes, the defined length of the measurement period in this experiment. The WG runs another few minutes before the first thread finishes, after it repeated the task chain 6 times.

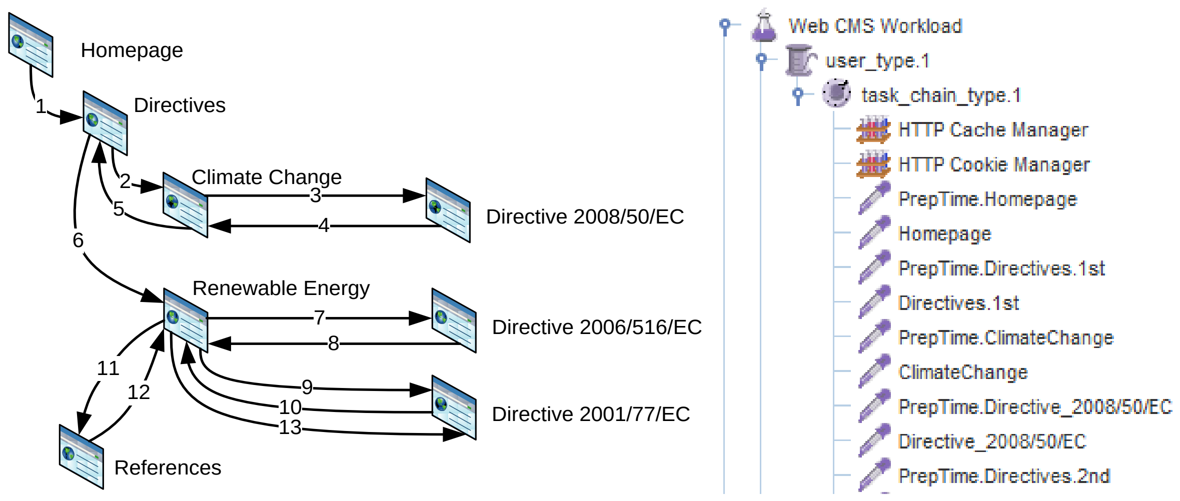

Figure 2
Structure of the website (arrows and numbers indicate the order of the steps in the task chain) and the Apache JMeter test plan that realizes the workload

The evaluation of the measured power data can be interpreted as follows: On average, Joomla CMS with hard disk cache consumes less energy (N=30, mean = 31.009 Wh, standard deviation = 0.096 Wh) than Joomla CMS without hard disk cache (N=30, mean = 33.937 Wh, standard deviation = 0.163 Wh). This difference was significant t(47.113) = 84.575, $p < 0.01$ (equal variances not assumed).

For the workload presented above, using a hard disk cache saves approx. 8.6% of electrical energy. This may be further increased by activating Joomla's support for browser caches, so that browsers do not request a web page again on subsequent requests.

4.5 Measuring Software on Desktop PCs

First, as was already described for servers above, a clean setup of the SUT is necessary. Furthermore, it is important to deactivate services that may bias the measurement, e.g. file system search indexers or memory resistant antivirus software. An appropriate WG may be a desktop automation tool, which records and replays mouse and keyboard actions. These tools usually implement three components: a macro recorder, a macro editor, and a macro executor. The macro recorder is used to record users' interactions out of real mouse movements and key strokes. The macro editor is used to adjust these macros, i.e. to adjust preparation times, to insert wait times after key strokes or mouse clicks, or to insert log statements for the validation of the workload. The macro executor executes the macros on the SUT. The measurement may be biased by the WG and the performance data recorder, which are both executed on the SUT.

4.5.1 Experimental Setup

The hardware basis of our desktop tests is an Asus motherboard (P4B533-V) with one Intel Pentium 4 single core CPU (@2.4 GHz, 1 GB RAM, 20 GB HDD). We decided to use Windows 7 in our test setup, because Windows XP, which is still the most common operating system, is phased out and Windows 7, the second most common operating system, rises steadily (StatCounter 2011b). In order to record performance statistics, we used Windows' Performance Monitor.

For workload generation, we used MouseRobot (http://www.automationbox.com/), a desktop automation tool. Besides the basic requirements of a WG, it also detects user interface (abbr. UI) objects of Windows applications, e.g. the address bar of web browsers or text elements. Thus, it is independent of absolute mouse positions, which simplifies maintenance of macros, especially if the UI structure of the tested applications changes slightly. Unfortunately, it is not capable of generating normally distributed random preparation times. As a viable solution, we defined constant preparation times.

4.5.2 Exemplary Measurements and Results with Web Browsers

As an example for software that is run on desktop PCs we compared the energy consumption while browsing the Web with the two most popular web browsers *Microsoft Internet Explorer* (9.0.8112.16421IC) and *Mozilla Firefox* (4.0.1) (StatCounter 2011a). Since the power consumption depends on the type of content shown in the browser (Walton 2009, Kern/Dick/Johann 2011), we defined test scenarios for websites with different types of content. The scenarios were designed for a duration of approx. 10 minutes test run, which means that basic steps were repeated several times.

The main content of the knowledge base *Wikipedia* (http://en.wikipedia.org/) is simple text and images. For testing, we simulated a typical user behavior while searching for a specific article. After loading the article, the reading process is simulated by scrolling down to a specific position, waiting there for two minutes, and then following a hyperlink to another article at the end of the page.

The evaluation of the measured power data can be interpreted as follows: On average, *Firefox* on *Wikipedia* consumed less energy (N = 30, mean = 9.382 Wh, standard deviation = 0.111 Wh, mean of duration = 00:08:13, standard deviation of duration = 00:00:04) than *Internet Explorer* (N = 30, mean = 10.771 Wh, standard deviation = 0.145 Wh, mean of duration = 00:09:05, standard deviation of duration = 00:00:06). This difference was significant t(58) = -41.544, $p < 0.01$ (equal variances assumed). In this scenario, the average energy conservation of *Firefox* compared to *Internet Explorer* is about 12%.

As a website, which uses a lot of JavaScript, we chose *Google Maps* (http://maps.google.com/). The testing scenario includes searching for a particular city, interacting with the viewed map, and searching for a route from the initial location to the focused location. Afterwards the directions are printed to a file.

The evaluation of the measured power data can be interpreted as follows: On average, *Firefox* on *Google Maps* consumed less energy (N=30, mean = 11.871 Wh, standard deviation = 0.244 Wh, mean of duration = 00:08:39, standard deviation of duration = 00:00:11) than *Internet Explorer* (N=30, mean = 14.788 Wh, standard deviation = 1,434 Wh, mean of duration = 00:10:35, standard deviation of duration = 00:01:07). This difference was significant t(30.686) = -10.981, $p < 0.01$ (equal variances not assumed). In this scenario, the average energy conservation of *Firefox* compared to *Internet Explorer* is about 19%.

The results show a dependency of the selection of software and electrical power consumption. Indeed, it depends on the scenario which software product outperforms the other regarding energy efficiency. In order to give general advice to users, it is necessary to gain more knowledge on real user behavior and types of usage, so that the workloads can be modeled more realistically.

4.5.3 Problems in Measuring

During the measurement, the WG and the Performance Monitor are executed on the SUT. It needs to be taken into account that this affects the measurement. Additionally, the duration of MouseRobot's UI object detection slightly differs depending on the complexity of the UI of the automated application program, which biases the measurement. Hence, this feature should be used sparingly or another WG should be used.

We used real websites and not partial copies or mock-ups. This has a huge influence on the measurement results, i.e. if the content changes unexpectedly during the experiments. Consider, e.g. graphical advertisements that are substituted more or less regularly with new images, animations, or even videos. Hence, if the variance of the response times is too large or if the presented content is unstable, one should choose a local mock-up that serves the content necessary for conducting the tests.

5. Conclusion and Outlook

In our paper we described a systematical approach on how to measure the energy consumption of software. Our test rig contains the tested system and software itself, some modules for generating workload during the test, a power meter, and an environment for evaluating the collected data. Exemplary tests show the influence of software usage on the consumption, even if the energy management of operating system is probably the predominant driver for the resulting energy consumption. Our method differs for desktop and server based software. For both cases, it applies realistic workloads, which are expected to occur in real life application areas. In this point, it differs significantly from other methods, which often use short running workloads that do not represent real user behavior. Thus, our method can be used to support software developers during software development, purchasers when evaluating tenders for software products, as well as administrators and users in making informed decisions on software products they plan to use.

Acknowledgement

This paper evolved from the research and development project "Green Software Engineering" (GREENSOFT), which is sponsored by the German Federal Ministry of Education and Research under reference 17N1209. The contents of this document are the sole responsibility of the authors and can under no circumstances be regarded as reflecting the position of the German Federal Ministry of Education and Research.

Bibliography

Albertao, F., Xiao, J., Tian, C., Lu, Y., Zhang, K.Q., Liu, C. (2010): Measuring the Sustainability Performance of Software Projects, in: IEEE Computer Society (ed.): 2010 IEEE 7th International Conference on e-Business Engineering (ICEBE 2010), Shanghai, China, pp. 369–373

Amsel, N., Tomlinson, B. (2010): Green Tracker: a tool for estimating the energy consumption of software, in: Association for Computing Machinery (ed.): CHI EA '10: Proceedings of the 28th international conference on Human factors in computing systems, New York, pp. 3337–3342

Capra, E., Formenti, G., Francalanci, C., Gallazzi, S. (2010): The Impact of MIS Software on IT Energy Consumption, http://web.up.ac.za/ecis/ECIS2010PR/ECIS2010/Content/Papers/0073.R1.pdf, 2010-10-25

Coroama, V., Hilty, L.M. (2009): Energy Consumed vs. Energy Saved by ICT – A Closer Look, in: Wohlgemuth, V., Page, B., Voigt, K. (eds.): EnviroInfo 2009: Environmental Informatics and Industrial Environmental Protection: Concepts, Methods and Tools, proceedings of the 23rd International Conference Environmental Informatics - Informatics for environmental protection, sustainable development and risk management, September 09 - 11, 2009, HTW Berlin, University of Applied Sciences, Germany, Aachen, pp. 353–361

Dick, M., Naumann, S. (2010): Enhancing Software Engineering Processes towards Sustainable Software Product Design, in: Greve, K., Cremers, A.B. (eds.): EnviroInfo 2010: Integration of Environmental Information in Europe, Proceedings of the 24th International Conference on Informatics for Environmental Protection, October 6 - 8, 2010, Cologne/Bonn, Germany, Aachen, pp. 706–715

Dick, M., Naumann, S., Kuhn, N. (2010): A Model and Selected Instances of Green and Sustainable Software, in: Berleur, J., Hercheui, M.D., Hilty, L.M. (eds.): What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience, 9th IFIP TC 9 International Conference, HCC9 2010 and 1st IFIP TC 11 International Conference, CIP 2010, Held as Part of WCC 2010, Brisbane, Australia, September 20-23, 2010. Proceedings, Berlin, Heidelberg, pp. 248–259

Dirlewanger, W. (2006): Measurement and rating of computer systems performance and of software efficiency, An introduction to the ISO/IEC 14756 method and a guide to its application, Kassel

Erdmann, L., Hilty, L.M., Goodman, J., Arnfalk, P. (2004): The Future Impact of ICTs on Environmental Sustainability, Technical Report EUR 21384 EN, Casal, C.R., Wunnik, C.V., Sancho, L.D. (et al.) (eds.), European Commission; Joint Research Centre; IPTS - Institute for Prospective Technological Studies, Seville, http://ftp.jrc.es/EURdoc/eur21384en.pdf, 2011-07-26

Field, A.P. (2009): Discovering statistics using SPSS, 3. ed., Los Angeles

Global e-Sustainability Initiative; The Climate Group (2008): SMART 2020: Enabling the low carbon economy in the information age, http://www.smart2020.org/_assets/files/02_Smart2020Report.pdf, 2011-07-26

Göhring, W. (2004): The Memorandum "Sustainable Information Society", in: Minier, P., Susini, A. (eds.): Sh@ring, proceedings of the 18th International Conference Informatics for Environmental Protection, EnviroInfo 2004, October 21-23, 2004, CERN, Geneva, Genève, pp. 278–286

Hilty, L.M., Köhler, A., von Schéele, F., Zah, R., Ruddy, T. (2006): Rebound effects of progress in information technology. In: Poiesis & Praxis: International Journal of Technology Assessment and Ethics of Science, volume 4, number 1, pp. 19–38

International Standard ISO/IEC 14756:1999: Information technology -- Measurement and rating of performance of computer-based software systems, 2010-06-17

Kern, E., Dick, M., Johann, T., Naumann, S. (2011): Green Software and Green IT: An End User Perspective, in: Golinska, P., Fertsch, M., Marx-Gómez, J. (eds.): Information Technologies in Environmental Engineering, Proceedings of the 5th International ICSC Symposium on Information Technologies in Environmental Engineering (ITEE 2011), Berlin, pp. 199–211

OECD (2010): OECD Information Technology Outlook 2010, Paris

Smith, C.U., Williams, L.G. (2002): Performance solutions, A practical guide to creating responsive, scalable software, Boston

Sommerville, I. (2011): Software engineering, 9. ed., Boston

Top 5 Browsers from Jun 10 to Jun 11 (2011a), StatCounter, http://gs.statcounter.com/#browser-ww-monthly-201006-201106, 2011-07-21

Top 5 Operating Systems from Jun 10 to Jun 11 (2011b), StatCounter, http://gs.statcounter.com/#os-ww-monthly-201006-201106, 2011-07-20

Strange, T., Bayley, A. (2008): Sustainable development, Linking economy, society, environment, Paris

Tayeb, J., Bross, K., Chang, B.S., Cong, L., Rogers, S. (2010): Intel® Energy Checker Software Development Kit User Guide, Intel Corporation (ed.), http://software.intel.com/file/32957/, 2011-07-26

United Nations General Assembly (1987): Report of the World Commission on Environment and Development, Our common future, UN document no. A/42/427 English, New York

Walton, J. (2009): Browser Face-Off: Battery Life Explored, http://www.anandtech.com/show/2834/1, 2011-07-26