Towards an Integrated Full-Stack Green Software Development Methodology

Philippe Roose
LIUPPA, E2S, University of Pau, France
Philippe.Roose@univ-pau.fr

Sergio Ilarri
I3A, University of Zaragoza, Zaragoza, Spain
silarri@unizar.es

Jorge Larracoechea
LIUPPA, E2S, University of Pau, France and University of Zaragoza, Spain
jorge-andres.larracoechea@etud.univ-pau.fr

Yudith Cardinale
Universidad Simón Bolívar, Caracas, Venezuela
ycardinale@usb.ve

Sébastien Laborie
LIUPPA, E2S, University of Pau, France
Sebastien.Laborie@iubayonne.univ-pau.fr

Abstract

Existing green/eco responsible approaches for IT are frequently domain-specific and very focused on one topic. For example, some works are focused on saving energy with better virtual machine management on cloud infrastructures or data management in wireless sensor networks, in order to minimize the data transfers and sensors' wakeups. Nevertheless, they consider only limited aspects in the whole software development process; indeed, very few researches propose a global approach. In this context, we envision a green development methodology that approaches energy saving aspects from the design phase and at all the system layers (software, hardware, user requirements, execution contexts, etc.), which can provide positive leverage as well as avoid side effects (one decision can be positive at one system layer but may trigger negative impact on other layers). We motivate the interest of this vision and describe key ideas regarding how to address these considerations in the development methodology.

Keywords: eco-responsible and green computing, holistic approach, green software development methodology, green data management

1. Introduction

All studies related to technology converge towards the same conclusion: Information Technologies (IT) will have no other choice but to be sustainable. The fast growth of the digital sector provides new paradigms and activities, leading to growth and employment in every economic sector. However, this phenomenon also raises a number of questions, as information technologies are also the cause of significant environmental impacts at all the stages of the lifecycle of IT. The IT resources from the cloud seem to be unlimited, but we certainly know that those from our planet are not. For instance, every year, the "Earth Overshoot Day" is occurring earlier (in 2020, it was the 22nd of August). Therefore, emerging IT technologies, existing practices, and algorithms need to be redefined for energy-efficient, energy-proportioned, and sustainable operations [17]. An increasing amount of research studies are now dedicated to integrating green/eco responsible approaches in order to save energy, when using IT.

Currently, most *GreenIT* research focuses on hardware manufacturing efficiency (recycling, materials used, etc.), and during the last few years some scientists have tackled

the problem of decreasing the energy consumption footprint of data centers and middleware by using more virtualization, load balancing techniques, and image migration [3], [10], [18]. At a lower scale, some computer scientists focus on the use of languages and how to develop code and implement numerical services [16]. A high-level interpreted language such as PHP or Python consumes much more than a low-level and compiled language [13]. A program is, of course, code, but also the result of applying a software engineering approach. Our hypothesis is that an eco-responsible software engineering approach will strongly benefit the energy consumption of applications. If an application (including its interactions) is well designed (i.e., eco-responsibly designed), then we can increase and optimise the performance-energy consumption ratio while satisfying the users' and application's needs, by using autonomic techniques.

Green Computing (part of GreenIT) is a recent topic in computer science, especially when compared to other traditional topics in this domain. From the software perspective, most research has focused on languages (for example, PHP7 consumes 64% less than the PHP6 release) and programming techniques (as the GreenSoft project [10] does) and on data centers and middleware (virtual machine migration or consolidation, for example).

In this context, we envision an improvement of traditional development methodologies by considering energy savings during the whole development process, from the design phase. Our hypothesis is that a holistic approach, which includes green and eco-responsible concerns at all the system development stages can provide positive leverage as well as avoid side effects (one decision can be positive at one system layer but may trigger negative impact on other layers). The main idea is to consider energy consumption aspects related to hardware, software, user requirements, scalability, data and service placement, as well as deployment policies (duplication, [re-]deployment, etc.), during the whole cycle of the development process. In this paper, we motivate the interest of this vision and describe key ideas regarding how to address these considerations in the development methodology. The structure of the rest of this paper is as follows. In Section 2, we review some relevant related work. In Section 3, we describe the key ideas of our proposal. Finally, in Section 4, we summarize our conclusions and describe ideas for prospective research.

2. Related Work

Literature on Green IT can be analysed through different lens: (i) the working level (or focus); (ii) the life cycle position; and (iii) the working techniques. Firstly, state-of-the-art literature works focus on saving energy at different working levels, including *hardware (H)*, *software components (C)*, software architecture or *components assemblage (A)*, and *software deployment (D)*. Secondly, existing proposals consider different steps of the software life cycle, from designing and coding till software execution (running), as well as the methodology guidance. Finally, we may consider the multiple techniques applied on the proposed works, including active monitoring, definition of SLAs (Service Level Agreements), or "greenness" indicators. Several proposals in the literature have considered energy savings on the hardware level or on software deployment [4], [15], but less efforts have been invested regarding software components [16] and architectures [9]. Similarly, multiple efforts could be observed concerning software execution, mainly in data centers/cloud infrastructures [3,4], [18], while we find less efforts on software design and methodologies [6], [9], [16]. Besides, most works focus on large infrastructures, such as HPC clusters, data centers, and cloud infrastructures [2], [4], [15], [18], while only a few works consider mobile applications; besides, in the few proposals tackling mobile apps, the proposed solutions often consider only offloading the application execution into cloud infrastructures [17]. For instance, the study presented in [9] considers sustainability as a quality attribute, following the ISO 25010 format. The authors propose a set of quality properties concerning the consumption of resources. The software architecture is considered through different viewpoints. Each viewpoint is associated with a set of issues, identified by key questions (e.g., how can we measure energy consumption on the different nodes executing the software?). An energy profile is considered to support possible adjustments on software deployment. Unfortunately, even if a catalogue of best practices is mentioned in [16], no guidelines are supplied for helping software developers to define

these profiles or support them in how to appropriately use or measure quality attributes and properties. Similarly, the GreenSoft project [10] advocates a holistic approach for green software development, including the development as well as the maintainability, uninstallation, and recycling aspects. Unfortunately, this project focuses mainly on hardware issues and energy consumption measurements, with small attention to the software development. The development phase considers only the environmental costs (how much energy is needed to power the workstations, the cost for remote work, etc.), focusing on a "software-hardware" matching (by comparing the consumption of different software instances), or on the identification of hotspots in the source code, but ignoring the impact of software development decisions. Furthermore, the authors analyse the energy consumption of multiple Java I/O (input/output) APIs, using different parameters, and demonstrate that significant energy savings are possible. Indeed, since I/O operations are often used millions of times in data centers, choosing the right API with an appropriate buffer size for a particular operation can lead to energy savings. Energy savings in data centers/cloud servers is also considered in [4], whose authors claim that eco-efficiency metrics might be part of a multi-criteria optimization process with both performance and environmental friendliness goals.

Several authors underline the challenges to estimate the energy consumption of software applications and components [4], [9], [12]. Not only quantifying the energy consumption of individual components in a complex software application is a challenging task, but also precisely measuring the energy consumption often demands special equipment for measuring the hardware consumption. Whilst such equipment could be affordable in large infrastructures [6], it is not the case for mobile or edge devices, which means that energy consumption monitoring is a challenging task for interactive applications. Some authors tackle this issue by proposing software-oriented estimations for energy consumption [9], [12]. Even if some authors suggest that common performance metrics may provide insights into the behaviour of hardware components [9], and consequently on the energy consumption behaviour, the use of such metrics is not integrated in a full design methodology. Thus, no guidance is proposed for software developers during application design. Developers are left alone regarding the challenging task of applying such indicators on their software application. This lack of guidance may lead to energy wasting during the software lifetime.

Another important perspective is the scalability. If we can imagine various deployments to save energy, determining the optimal deployment is not always possible for 'big' applications composed of many components (well known as an NP-Complete problem). Besides, different authors underline the existence of a trade-off between the energy consumption, the application performance, and the user's satisfaction [4]. A trade-off between the development efficiency and the energy efficiency is also suggested; the authors conclude that different designs can have a significant impact on the energy efficiency of the software applications. However, reaching such optimum design demands guiding the software developers during the whole software engineering process.

3. Towards an Integrated Energy-Aware Software Development Methodology

Up to the authors' knowledge, no full-fledged eco-responsible software methodology exists. Even if a holistic view is recommended by some authors [10], no methodology covering the software design, deployment and execution has been proposed so far. This lack is currently observed by practitioners, as highlighted in [5], representing an important obstacle for green software development. If we want to increase the energy savings and propose the best possible deployment, we need a specific software engineering approach and ensure that the energy efficiency concerns do not impose a heavy burden on the developers. Self-adaptive systems are based on a MAPE (Monitoring, Analysis, Planning and Execution) approach [11]. The *Monitoring phase* needs capturing data. Such data are organized and structured in an information system called *Knowledge*, which is the K in the MAPE-K (Monitor-Analyze-Plan-Execute over a shared Knowledge) denomination. This knowledge database contains data about sensors, adaptive rules, application state, metrics,

etc. The current state of an application is called *Context*, which is why many research works associate context-aware computing and autonomic computing. Even if software applications can take advantage of context awareness to save energy [14], this behaviour is not commonly observed in the literature. To apply autonomic computing for saving energy, we need to tackle some challenges, notably concerning the *Monitoring phase*. Even though some works use context [11], and more generally domain information, in cloud computing systems in order to minimize financial costs, very few research works, such as [15], do it to minimize energetic costs, and only at the data center level (none of them at application level). Covering this gap is one of our perspectives in our work.

A traditional software engineering approach can be described as depicted in Figure 1, with eventually a back arrow (to form a feedback loop) between steps 3 and 4 (inspired by MAPE-K approaches).

Figure 1: Traditional software engineering approach

Without proposing a rupture of the classical software engineering approach, we argue that we can strongly improve energy savings by acting at each step, as illustrated in Figure 2. Even though, for simplicity, we illustrate a traditional development methodology, this approach integrating energy concerns in all the development phases is fully compatible with agile methodologies.

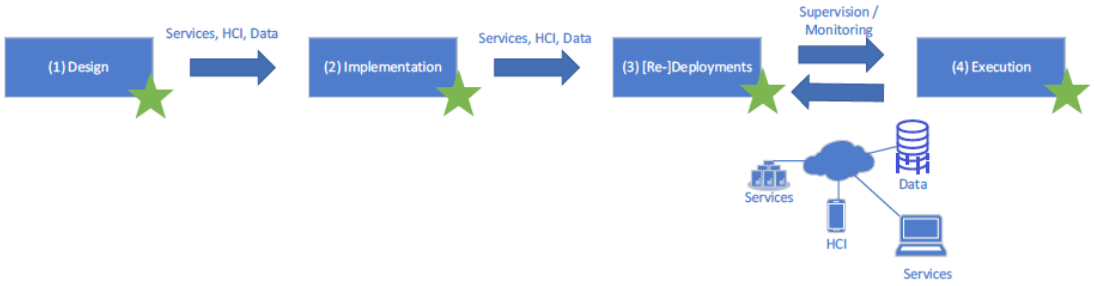

Figure 2: Software engineering approach enriched with energy saving concerns from the design phase

1) *Design* (including the *Analysis of Requirements*). The requirements step and the design of applications may involve three main aspects: functional and non-functional aspects (including functional and non-functional requirements), HCI (Human Computer Interactions) aspects, and data. It is now admitted that the design of an interface has an impact on the way it is used and exploited/consumed (e.g., depending on the data refreshment frequency required and the colour choices). Besides, the organisation of data (type of data, completion, synchronization, etc.) and of course the functional analysis (granularity, access mode, duplication/migration capability, running time, etc.) can also lead to significant differences in terms of energy consumption. Additionally, the usage and the type of functions have a strong impact. The energy debt has a higher influence (and needs additional attention) when there is an intensive use. The energy cost depends on the required processes; for example, a pure computation process has usually less impact on energy consumption than a streaming one (due to the high cost of data transfer). The identification of such information may help to focus on the correct aspects.

2) *Implementation*. As presented in Section 1, programming languages (including patterns and frameworks) have a strong impact on energy consumption. Although less visible, some other aspects such as the choice of the type of variables/objects (e.g., using *int* as opposed to using *integer*), control structures, and specific instructions also influence the energy cost of the *code's* execution. Current compilers can optimize the code, but from an execution time point of view instead of an eco-responsible point of view; some works analyse the impact of choosing among different compiler optimization levels, but a complete integrated and automated approach considering both metrics (performance and energy consumption), and analysing the potential trade-offs between them, is still missing.

3) *Deployment and redeployments*. The choice of deployment plays also an important

role. For example, a host using 20-80% of the CPU (on average) consumes more or less the same [7]. It may then be interesting to move some processes to other nodes in order to free some CPU under 20% (economy mode, to decrease the voltage consumption) or to avoid the burst mode (CPU consumption higher than 80%, where there is a strong *consumption* increase). Moreover, regarding the edge computing/WSN (Wireless Sensor Networks) philosophy, it may be interesting to move/duplicate some data to specific locations to enhance the data locality during execution (i.e., move the computation closer to the data) and thus minimize the transfer of data and the communication latency.

4) *Execution*. Measuring both the hosts' and apps' activity (HCI/services/data accesses) provides an important number of indicators (indicators on bandwidth consumption/services and hosts, detailed CRUD —Create, Read, Update and Delete— accesses, data *consumers*/providers, localizations, etc.). The analysis of those indicators, as done for context-aware apps and in autonomic computing, allow to have high level indicators to be re-injected into the decision process in order to better perform the deployments (learning techniques can also be applied to detect habits).

The loop between stage (3) and (4) implements this autonomic point of view. The objective here is to analyse the deployments and see how we can better deploy (or re-deploy) tasks in order to better use the hardware and avoid for example a CPU power boost phase, minimize swapping, or better use the network interfaces [1]. Such dynamic in tasks placement allows to move them to the most suitable devices in order to minimize/optimize the use of hardware resources. Actions on data with partial duplication or data migration, for example, is also a promising approach. Of course, those actions have to take into account the cost of moving/duplicating tasks/data, as well as the cost of keeping the data replicas synchronized (in case updates are allowed). We believe that the adoption of holistic approaches (encompassing all the stages of development, starting from the analysis, and including the design, deployment, execution, and reconfigurations steps) is the only way to really obtain significant gains in terms of energy savings.

4. Conclusions and Future work

In this paper, we have proposed an integrated energy-aware software development methodology considering all the system layers, from the design to the execution phase (thus addressing aspects related to the software, hardware, user requirements, data, and execution contexts). Our vision is to guide the designers as well as the software architects by providing useful metrics and parameters that then will be integrated in autonomic decisions to obtain energy savings. Furthermore, our approach can also integrate measures to raise the awareness of the users about their applications' energy footprint. The parameters and metrics will be integrated in the software development at the design time, which requires a middleware platform that respects those indications during execution, mainly based on a MAPE-K strategy.

To test and validate our methodology, we will develop a middleware that will integrate an eco-responsible usage and autonomic rules with dedicated metrics encompassing both green computing aspects and interactivity aspects. The result is a prototype under validation for technological transfer (with one patent under redaction). We are currently identifying key elements in software engineering in order to help designers to propose eco-responsible software and identify metrics to help by improving the placement of their execution (including the [re-]placement of data).

Acknowledgements

This work is partially supported by the University of Pau, E2S, I-Site Label of excellence, projects PID2020-113037RB-I00 and TIN2016-78011-C4-3-R (AEI/FEDER, UE), and the Government of Aragon (Group Reference T64_20R, COSMOS research group).

References

1. Alvarez Valera H. H., Dalmau M., Roose P., Larracoechea Gonzalez J. A., Herzog C.: An energy saving approach: Understanding microservices as multidimensional entities in P2P networks, 36th Annual ACM Symposium on Applied Computing (ACM SAC 2021), March 22-March 26(2021)
2. Beloglazov A., Abawajy J., Buyya R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," Future Generation Computer Systems, special section on "Energy efficiency in large-scale distributed systems", vol. 28, no. 5, pp. 755 – 768 (2012).
3. Botero J. F., Hesselbach X., Duelli M., Schlosser D., Fischer A., de Meer H.: Energy efficient virtual network embedding, IEEE Communications Letters, vol. 16, no. 5, pp. 756–759, May (2012).
4. Bunse C., Klingert S., Schulze T.: GreenSLAs: Supporting energy-efficiency through contracts, in Energy Efficient Data Centers, J. Huusko, H. de Meer, S. Klingert, and A. Somov, First International Workshop on Energy Efficient Data Centers (E2DC 2012), Lecture Notes in Computer Science (LNCS), vol 7396, Springer Berlin Heidelberg, pp. 54-68 (2012)
5. Chitchyan R., Becker C., Betz S., Duboc L., Penzenstadler B., Seyff N., Venters C.: Sustainability design in requirements engineering: State of practice, 38th International Conference on Software Engineering (ICSE 2016) Companion, New York, NY, USA, ACM, pp. 533–542 (2016)
6. Feng W.,: The Green Computing Book: Tackling Energy Efficiency at Large Scale, CRC Press, ISBN 9781439819876 (2014)
7. Ge C., Sun Z., Wang N.: A Survey of Power-Saving Techniques on Data Centers and Content Delivery Networks, IEEE Communications Surveys & Tutorials, vol. 15, issue: 3 (2012)
8. Insaurralde C. C.: Autonomic computing technology for autonomous marine vehicles, Ocean Engineering, volume 74, 1 pp. 233-246, december (2013)
9. Jagroep E., van der Werf J. M., Brinkkemper S., Blom L., van Vliet R.: Extending software architecture views with an energy consumption perspective, Computing, June (2017)
10. Mansouri K., Alti A. Roose P.: Selection and Composition of Cloud Smart Services Using Trust Semantic-Based Green Quality Approach, Third International Conference on Pattern Analysis and Intelligent Systems (PAIS 2018), Tebessa, Algeria, October (2018)
11. Kern E., Hilty L. M., Guldner A., Maksimov Y. V., Filler A., Groger J., Naumann S.: Sustainable software products - towards assessment criteria for resource and energy efficiency, Future Generation Computer Systems, vol. 86, pp. 199–210 (2018)
12. Noureddine A., Bourdon A., Rouvoy R., Seinturier L.: A Preliminary Study of the Impact of Software Engineering on GreenIT, in First International Workshop on Green and Sustainable Software, Zurich, Switzerland, June (2012).
13. Sabharwal M., Agrawal A., Metri G.: Enabling green IT through energy-aware software, IT Professional, vol. 15, pp. 19–27, 01 (2013)
14. Pereira R., Couto M., Ribeiro F., Rua R. Cunha J., Fernandes¨J. P., Saraiva J.: Energy Efficiency across Programming Languages - How Does Energy, Time, and Memory Relate ?, International Conference on Software Language Engineering (SLE 2017), Vancouver, BC, Canada, October 23–24 (2017)
15. Salomie I., Cioara T., Anghel I., Moldovan D., Copil G., Plebani P.: An energy aware context model for green IT service centers, in Service-Oriented Computing, E. M. Maximilien, G. Rossi, S.-T. Yuan, H. Ludwig, and M. Fantinato, International Conference on Service-Oriented Computing (ICSOC 2010), Lecture Notes in Computer Science (LNCS), vol 6568, Springer Berlin Heidelberg, 2011, pp. 169–180 (2010)
16. Singh J., Naik K., Mahinthan V.: Impact of developer choices on energy consumption of software on servers, Procedia Computer Science, vol. 62, pp. 385 – 394, 2015, 2015 International Conference on Soft Computing and Software Engineering (2015)
17. Shuja J., Ahmad R. W., Gani A., Abdalla A. I., Siddiqa A., Nisar K., Khan S. U., Zomaya A. Y.: Greening emerging IT technologies: techniques and practices, Journal of Internet Services and Applications, vol. 8, no. 1, p. 9, Jul 2017 (2017)
18. Valentini G. L., Lassonde W., Khan S. U., Min-Allah N., Madani S. A., Li J., Zhang L., Wang L., Ghani N., Kolodziej J., Li H., Zomaya A. Y., Xu C.-Z., Balaji P., Vishnu A., Pinel F., Pecero J. E., Kliazovich D., Bouvry P.: An overview of energy efficiency techniques in cluster computing systems, Cluster Computing, vol. 16, no. 1, pp. 3–15, March 2013 (2013)