MAP-I
Programa Doutoral em Informática

Green Software Engineering

Unidade Curricular em Paradigmas de Programação
Programming Paradigms
(UCPC)

UMinho, FEUP

October 2022

Resumo

This document describes a Ph.D. level course, corresponding to a Curriculum Unit credited with 5 ECTS. It corresponds to a joint UMinho-FEUP proposal for UCPC (Programming Paradigms) in the joint MAP-i doctoral program in Informatics, organized by three Portuguese Universities (Minho, Aveiro, and Porto).

LECTURING TEAM

UMinho:	João Saraiva
FEUP:	João Paulo Fernandes
Roskilde University, Denmark	Maja Hanne Kirkeby (invited lecturer: confirmed)
Coordinator:	João Saraiva

A. Programmatic Component

1. Theme, Justification and Context

Motivation

The world is increasingly aware of and concerned about sustainability and the green movement. Computers and their software play a pivotal role in our world, thus they have a special responsibility for social development and the welfare of our planet. In this century, the situation is becoming critical since software is everywhere! The widespread use of computer devices, from regular desktop computers, to laptops, to powerful mobile phones, to consumer electronics, and to large data centers is changing the way software engineers develop software. Indeed, in the forthcoming era of Artificial Intelligence (AI), Internet of Things (IoT) and edge computing, there are new concerns which developers have to consider when constructing software systems. While in the previous century both computer manufacturers and software developers were mainly focused in producing very fast computer systems, now energy consumption is becoming the main bottleneck when developing such systems [1].

Only recently has the software engineering community started performing research on developing energy efficient software, or green soft- ware. This is shadowed when compared to the research already produced in the computer hardware community. While research in green software is rapidly increasing, several recent studies with software engineers show that they still miss techniques, knowledge, and tools to develop greener software. Indeed, all such studies suggest that green software should be part of a modern Computer Science Curriculum.

1 Green Software Engineering: Course Content

The green software engineering module is a multidisciplinary course, combining several software engineering techniques and principles, namely:

Source Code Analysis and Transformation: In order to analyze and transform software systems we introduce two powerful source code manipulation techniques: Strategic and Aspect Oriented Programming. Strategic programming is a generic tree traversal technique that allows for expressing powerful abstract syntax tree analysis and transformations. Aspect oriented programming is introduced to allow developers to instrument the base source code

without adding the energy monitoring intrusive code, but keeping it in one aspect that is later weaved to the base program.

Green Aspect: In order to monitor the energy consumption, students need to traverse and instrument the source code with calls to APIs providing energy measurements at runtime. In our course, we consider two types of measurements: energy estimation provided by manufacturers of the CPUs, namely the RAPL framework developed by Intel [2, 3], or using hardware with energy sensors, like for example the ODroid hardware board[1].

Source Code Smells and Metrics: Code smells represent symptoms of poor implementation choices when developing software. Code smells are not faults, they make program understanding difficult, and possibly indicate a deeper problem in the software. Software metrics are usually used to detect source code smells, for example, a too long method smell.

Green Aspect: In our module on green software we present a catalog of energy greedy programming practices for Java and Android [4]. This catalog can also be seen as a energy smell catalog, where software metrics can be used to detect such smells in the source code.

Program Refactoring: refactoring is a controlled source-to-source transformation technique for improving the design of an existing (source code) software system. Its essence is applying a series of small semantic-preserving transformations. Refactorings are usually associated with code smells: for each smell there is a refactoring that eliminates it.

Green Aspect: We associate refactorings to the catalog of energy smells so that students can use a green refactoring to eliminate red smells. Because the main focus of refactoring is to improve comprehensibility, several refactorings may negatively affect energy consumption. Students also analyze how refactorings available from Java IDEs affect energy consumption. The catalog of green refactorings for Java data structures supported by the jStanley tool [5].

Technical Debt: Technical debt describes the gap between the current state and the ideal state of a software system. The key idea of technical debt is that software systems may include hard to understand/maintain/evolve artefacts, causing higher costs in the future development and maintenance activities. These extra costs can be seen as a type of debt that developers owe the software system.

Green Aspect: In our module we introduce the concept of Energy Debt [6] as the amount of unnecessary energy that a software system uses over time, due to maintaining energy code smells for sustained periods.

[1] `http://www.odroid.com`

Software Testing and Benchmarking Infrastructures: Software testing aims at ensuring that a software system is defect free. We present the usual levels of testing: unit, integration, system, regression and beta testing. Automated test case generation and property based testing is also studied in this course. Code coverage and mutation-based testing is used to assess the quality of the test suite. Moreover, we use testing framework and benchmarks infrastructures, like Google's Caliper[2] in order to execute programs.

Green Aspect: To measure energy consumption, the source code needs to be executed with proper inputs. We use system testing, where the automated test case generation techniques produces *real* inputs of the program under testing.

Fault Localization: When a software systems fails running the defined/generated test suite, programmers need to locate the fault and fix it. Spectrum-based Fault Localization (SFL) relies on test cases to run the program, and it uses statistical methods to assign probabilities of being faulty to source code components (methods, classes, statements, etc).

Green Aspect: Abnormal energy consumption can be seen as a software fault. In our course we defined a variant of SFL to locate energy leaks in the source code: Spectrum-based Energy Leak Localization (SPELL) [7, 8]. Students can use it to locate such energy hot-spots in their software.

Automated Program Repair: The goal of automated program repair is to take a faulty program and a test suite, and automatically produce a patch that fixes the program. The test suite provides the correctness criterion in this case, guiding the repair towards a valid patch.

Green Aspect: SPELL adapts fault localization to the green software realm, while green refactorings eliminate red smells with the aim of improving the energy efficiency of programs. We combine these two techniques in order to automate the energy-aware repair of energy inefficient software systems [9, 10].

2 Green Software Engineering: Objectives and Learning Outcomes

The objectives of the green software module are:

- Be able to monitor and analyse the energy consumption of software systems.

[2] `http://code.google.com/p/caliper/`

- Become aware of the impact of programming practices on energy consumption.
- Become familiar with the research problems in the field of green software engineering.

At the end of this course, PhD students should be able

- to instrument, measure and analyze the energy consumption of software systems.
- to locate energy inefficient (source) code in a software systems.
- to optimize the energy consumption of greedy software systems.

3 Green Software Engineeering: Course's Structure

The module of green software is part of the software analysis and testing course. This course is one semester long, with 5 ECTS. It is a non-mandatory course, included in the first year (second semester) of the master program on Informatics Engineering at Minho University.

The students have 3 hours per week in the classroom: one hour in a seminar room, where all theories and techniques are presented. The remaining two weekly hours are laboratory classes where students have the chance to experiment the introduced techniques for software energy consumption. The evaluation consists of two components: an individual written exam, and a group project on analyzing and optimizing the energy consumption of a given software system. The considered software system is the students' project, developed in the introductory course to object oriented programming the semester before (by second year students). The idea is to provide students in the course with a simple, non fully optimized system.

In order to analyze and optimize the energy consumption of Java based software systems, we present the students a catalog of energy-greedy Java programming practices. The main goal is to make students aware of some features of Java's source code that may indicate an abnormal energy consumption by the software. The students are also presented with a possible solution by performing a refactoring of the source code into a more energy efficient one. Moreover, software tools that locate such features and (semi) automatically optimize the code are also presented. In laboratory sessions, the students are able to experiment with smell detection and optimization. Then, outside of class, students have to work in group (three students per group) and apply the catalog/tools in order to optimize the energy consumption of a real software project.

4 Green Software Engineeering: Supporting Tools

As reported in literature, there is a lack of tool support for sustainable software engineering [11]. It is hard to train students on how to write green software if we cannot measure the energy consumption of software. Therefore, easy-to-use tools that can provide detailed and accurate power measurement play a critical role in green computing and green software education. The lacking of such tools and of a low cost infrastructure that can support lab experiments in green software courses is clearly one of the key obstacles that prevent educators from introducing green computing/software topics to the CS curriculum.

Fortunately, as power consumption becomes increasingly important, most vendors today provide power measurement APIs such as Intel's Running Average Power Limit (RAPL) [2, 12, 3] and Nvidia's Management Library (NVML) [13]. These APIs provide functionalities to measure the real-time power consumption of CPU, DRAM, and GPU for programs running on desktop workstations or servers. Since power data is accessed through machine specific registers (MSRs), users may need special permission to log in an Linux OS to obtain such data, which is not trivial for some students (especially undergraduate students). A number of useful tools are also available for energy analysis of applications running on mobile devices. For example, the Android Qualcomm Trepn app [14] can profile hardware usage (GPS, Wifi, etc.), resource usage (memory, CPU), and the power consumption of both the system and any standalone Android app that run on a Snapdragon chipset. PETRA [15] is another model-based tool that can estimate the energy consumption of Android apps at a coarse-grained level. Hardware-based power measurement tools that provide high frequency and high precision profiling (e.g. Monsoon [16]) are also available but they are generally expensive.

Green Software Laboratory Tools: Under the umbrella of the Green Software Laboratory project [17, 18], several tools and other software artifacts (*i.e.*, software repositories) have been developed, which will support the laboratory classes of this course, namely:

- *SPELL* [8]: A toolkit to measure the energy consumption of a Java based program and detect potential energy hot spots through an adapted Spectrum-based Fault Localization technique.
- *jStanley* [5]: An Eclipse plugin that automatically refactors Java collections to more energy efficient ones.
- *E-Debitum* [19] - A SonarQube extension to manage the Energy Debt of Android-based software systems. This tool detects the Energy Greedy Android Patterns (EGAPs) presented in our energy smells ctalog for Android [4].

- *GreenSource* and *AnaDroid* [20]: A repository of android source code applications tailored for green software analysis and a tool to static analyze and dynamically monitor the energy consumption of such applications.
- *E-Manafa* [21] - A plug-and-play device-independent, model-based energy profiler for Android devices.

These tools have been developed by former students who motivated by the green software module of the course decided to perform their MSc and PhD thesis in this active research field.

5 Course Structure and Contents

This course is structured in three units: the first two present the foundations of source code analysis and transformations. In unit 3, we will present analysis techniques for debugging and fault localization in software systems. Finally, in unit 4, we will study analysis and transformations of source code in the context of embedded computing.

Course unit	Lecturer(s)
Energy Consumption: Monitoring	Maja Hanne Kirkeby
Energy Consumption: Analysis	João Paulo Fernandes
Energy Consumption: Fault Localization	João Saraiva
Energy Consumption: Optmization	João Saraiva & João P. Fernandes

4. Teaching Methods and Student Assessment

This course will consist of theoretical and practical components. In the theoretical component, a set of seminars will be delivered by the lecturing team, invited speakers, and the students themselves.

- The basic and advanced contents of this course will be presented by the lecturing team. We will be using lecturing material developed within research groups, namely in the context of a MSc course on Software Analysis and Testing with a Green Flavor offered at UMinho and the lecturing material produced in the context of the SusTrainable project: a Strategic Partnerships for higher education of the European Union coordinated by João Paulo Fernandes.
- Students will give one presentation of proposed research papers. The presentation will follow a standard conference talk model. The research papers will be proposed by the lecturing team, and they consist of a selection of recently published papers in conferences and journals in Green Software.

The practical component consists of the development of an individual project. This project requires the use of a specific software system for the monitoring, analysis and optimization of the energy consumption of a software system. The students are also expected to give a presentation, in the form of a tool demo, of the software system they decide to use. They are also supposed to write a report of the project as a research paper.

Student Assessment

During the this course the students will have to give a research talk, to develop a research project, to give a tool demo talk presenting the software system they adopt, and finally to write the project report as a scientific paper. Thus, the student will be assessed through these four activities.

5. Basic Bibliographic References

- Book: *Green in Software Engineering*, Coral Calero, Mario Piattini, Springer, 2015. https://doi.org/10.1007/978-3-319-08581-4
- Book: *Software Sustainability*, Coral Calero, M^a Ángeles Moraga, Mario Piattini, Springer, 2021. https://doi.org/10.1007/978-3-030-69970-3
- Book Chapter: *Patterns and Energy Consumption: Design, Implementation, Studies, and Stories*, D. Feitosa, L. Cruz, R. Abreu, J. P. Fernandes, M. Couto, J. Saraiva, Springer International Publishing, Cham, 2021, pp. 89–121. https://doi.org/10.1007/978-3-030-69970-3_5
- Tutorial: *Energy Consumption and Optimization of Software*, Maja H Kirbeby, COST ACTION CERCIRAS Summer School 2022, Split, Croacia, September 2022.
- Tutorial: *Energy Debt: Foundations, techniques and tools*, J. Saraiva, R. Rua, SusTrainable Summer School 2022, Rijeka, Croacia, July, 2022, Revised Selected Papers, Springer International Publishing, (to appear).
- CACM article: *Energy efficiency: a new concern for application software developers*, G. Pinto, F. Castor, Communications of the ACM 60 (12) (2017) 68–75. https://doi.org/10.1145/3154384

B. Lecturing Team

1. Team Presentation

This course is supported by a team involving researchers form both the University of Minho, School of Engineering (João Saraiva), and the University of Porto, FEUP (João Paulo Fernandes).

All team members are working, and have worked actively in the past few years, on topics that are directly related to the subjects covered by this course, as detailed below.

2. Coordinator

The coordinator of the unit is João Saraiva.

3. Short Presentation of Team Members

In the sequel we introduce a brief presentation of each team member, which includes, for each of them, up to 5 key publications related to the scientific area in which this course is proposed.

João Saraiva is an Associate Professor at the Departmento de Informática, Universidade do Minho, Portugal, and a research member of HASLab/INESC TEC. He obtained a Ph.D. degree in Computer Science from Utrecht University in 1999. His main research contributions have been in the field of programming language design and implementation, program analysis and transformation, and functional programming. He supervised 11 PhD projects (8 awarded and 3 running) and and has published over 100 publications (scopus h-index: 18). He served in over 90 PCs of international events, and in the evaluation committees of 7 national agencies: Austria, Belgium, France, Portugal, The Netherlands, Spain, and Uruguay.

He coordinated research projects both at national level with projects funded by FCT and QREN, and at international level with projects funded by EPSRC (UK), FLAD/NSF (USA), and by the EU. He is one of the founders of the successful series of GTTSE summer schools and was the organizing chair of ETAPS'07.

Key Publications:

- Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, João Saraiva, *Ranking Programming Languages by Energy Efficiency*. Journal of Science of Computer Programming, volume 205: 102609 (2021)
- Rui Pereira, Tiago Carção, Marco Couto, Jácome Cunha, João Paulo Fernandes, João Saraiva, *SPELLing out energy leaks: Aiding developers locate energy inefficient code*. Journal of Systems and Software, volume 161: 110463 (2020)
- João Saraiva, Ziliang Zong, Rui Pereira, *Bringing Green Software to Computer Science Curriculum: Perspectives from Researchers and Educators*. 26th ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2021), pages 498-504, ACM 2021.
- Marco Couto, Daniel Maia, João Saraiva, Rui Pereira, *On energy debt: managing consumption on evolving software*. Proceedings of the 3rd International Conference on Technical Debt (TechDebt@ICSE 2020), Seoul, Republic of Korea, pages 62-66, 2020.
- Rui Rua, Marco Couto, João Saraiva, *GreenSource: a large-scale collection of Android code, tests and energy metrics*. 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR 2019), pages 176-180, 2019.

Funded Projects in Green Software:

- *Sustrainable: Promoting Sustainability as a Fundamental Driver in Software Development Training and Education* (SusTrainable), projecto financiado pelo *Erasmus+ Programme KA203 - Strategic Partnerships for higher education of the European Union* (projecto 2020-1-PT01-KA203-078646). 2020-2023.
- *GreenSoftwareLab – Towards an Engineering Discipline for Green Software*, projecto Financiado pela FCT, Investigador Principal, contracto ref. PTDC/EEI-ESS/5341/2014. 2016-2020.

Supervision of (ongoing) PhD projects:

- Walter Lucas, *Transformações seguras de código para rejuvenescimento de software*, Departamento de Ciências da Computação, Universidade de Brasília, Brasil, início em 2020. Em curso. (co-supervisor, supervisor: Prof. Rodrigo Bonifácio))
- José Nuno Macedo, *Improve Software Quality by Combining Fuzz Testing and Spectrum-based Fault Localization*, PhD Student under FCT grant `SFRH/BD/08184/2021`, Departamento de Informática, Universidade do Minho, início em 2018. Em curso. (supervisor)

- Francisco José Ribeiro, *Automated Program Repair*, PhD Student under FCT grant SFRH/BD/144938/2019, Departamento de Informática, Universidade do Minho, início em 2018. Em curso. (supervisor)
- Rui Rua, *Green Software in the Large: Repository and Analysis*, PhD Student under FCT grant SFRH/BD/146624/2019, Departamento de Informática, Universidade do Minho, início em 2018. Em curso. (supervisor)

João Paulo Fernandes is Associate Professor at the Informatics Engineering Department of the Faculty of Engineering of the University of Porto, Portugal. His research is focused on the rigorous analysis and transformation of software, with the general goal of optimizing its non-functional properties while still ensuring its functional correctness.

For almost 10 years now, he is focused on optimizing the energy efficiency of software systems, an area in which he tries to provide developers with information and tools to support the development of energy efficient software and at the same time that he seeks to support end users, namely of mobile devices, in adopting more efficient usage patterns. In these contexts, he has founded/coordinate(d) and/or am involved in projects and initiatives such as GreenHub, Green Sw Lab, GreenHaskell and Sustrainable.

Currently, he is also a member of the Artificial Intelligence and Computer Science Laboratory of the University of Porto (LIACC). In the past, he has held different types of positions in several institutions such as the University of Minho (Monitor, 2002-2004), the Polytechnic Institute of Porto (Assistant Professor, 2008-2010), the University of Porto, Faculty of Engineering (Assistant Professor, 2010-2012), the University of Beira Interior (Assistant Professor, 2012-2016) and the University of Coimbra (Assistant Professor, 2016-2020).

He has graduated in Mathematics and Computer Science from the University of Minho, in 2004 (best of class, with an average score of 17/20), having conducted his graduation thesis under the PURe project. Later, in March 2009, he received a Ph.D. degree from the same university, following his work on the Design, Implementation and Calculation of Circular Programs.

Key Publications:

- Rui Pereira, Hugo Matalonga, Marco Couto, Fernando Castor, Bruno Cabral, Pedro Carvalho, Simão Melo de Sousa, João Paulo Fernandes: *GreenHub: a large-scale collaborative dataset to battery consumption analysis of android devices*. Empir. Softw. Eng. 26(3): 38 (2021)

- Wellington Oliveira, Renato Oliveira, Fernando Castor, Gustavo Pinto, João Paulo Fernandes: *Improving energy-efficiency by recommending Java collections*. Empir. Softw. Eng. 26(3): 55 (2021)
- Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, João Saraiva, *Ranking Programming Languages by Energy Efficiency*. Journal of Science of Computer Programming, volume 205: 102609 (2021)
- Rui Pereira, Tiago Carção, Marco Couto, Jácome Cunha, João Paulo Fernandes, João Saraiva, *SPELLing out energy leaks: Aiding developers locate energy inefficient code*. Journal of Systems and Software, volume 161: 110463 (2020)
- Marco Couto, João Saraiva, João Paulo Fernandes: *Energy Refactorings for Android in the Large and in the Wild*. SANER 2020: 217-228

Funded Projects in Green Software:

- *Sustrainable: Promoting Sustainability as a Fundamental Driver in Software Development Training and Education* (SusTrainable), projecto financiado pelo *Erasmus+ Programme KA203 - Strategic Partnerships for higher education of the European Union* (projecto 2020-1-PT01-KA203-078646). Principal Investigator, 2020-2023.
- *GreenSoftwareLab – Towards an Engineering Discipline for Green Software*, projecto Financiado pela FCT, contracto ref. PTDC/EEIESS/5341/2014. 2016-2020.

Supervision of (ongoing) PhD projects:

- Cláudio Gomes, *Quantum Computing applied to Sustainability*, University of Porto and CMU - USA , since September 2021. (supervisor)
- Bernardo Santos, *On the Energy Efficiency of Compiler Optimizations*, University of Porto, since September 2022. (supervisor)

Referências

[1] G. Pinto, F. Castor, Energy efficiency: a new concern for application software developers, Communications of the ACM 60 (12) (2017) 68–75. URL https://doi.org/10.1145/3154384

[2] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, C. Le, RAPL: memory power estimation and capping, in: International Symposium on Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE, IEEE, 2010, pp. 189–194.
URL https://doi.org/10.1145/1840845.1840883

[3] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, Z. Ou, RAPL in action: Experiences in using RAPL for power measurements, ACM Trans. Model. Perform. Eval. Comput. Syst. 3 (2) (mar 2018).
URL https://doi.org/10.1145/3177754

[4] M. Couto, J. Saraiva, J. P. Fernandes, Energy refactorings for android in the large and in the wild, in: 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2020, pp. 217–228.
URL https://doi.org/10.1109/SANER48275.2020.9054858

[5] R. Pereira, P. Simão, J. Cunha, J. Saraiva, jStanley: Placing a Green Thumb on Java Collections, in: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, ACM, New York, NY, USA, 2018, pp. 856–859.
URL http://doi.acm.org/10.1145/3238147.3240473

[6] M. Couto, D. Maia, J. Saraiva, R. Pereira, On energy debt: Managing consumption on evolving software, in: Proceedings of the 3rd International Conference on Technical Debt, TechDebt '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 62–66.
URL https://doi.org/10.1145/3387906.3388628

[7] R. Pereira, T. Carção, M. Couto, J. Cunha, J. P. Fernandes, J. Saraiva, Helping programmers improve the energy efficiency of source code, in: Proceedings of the 39th International Conference on Software Engineering Companion, ICSE-C '17, IEEE Press, 2017, p. 238–240.
URL https://doi.org/10.1109/ICSE-C.2017.80

[8] R. Pereira, T. Carção, M. Couto, J. Cunha, J. P. Fernandes, J. Saraiva, Spelling out energy leaks: Aiding developers locate energy inefficient code, Journal of System and Software. 161 (2020).
URL https://doi.org/10.1016/j.jss.2019.110463

[9] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Saraiva, Energyware analysis, in: 7th Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA), Vol. 2217, CEUR Workshop Proceedings, 2018.

[10] R. Pereira, Energyware engineering: Techniques and tools for green software development, Ph.D. thesis, Universidade do Minho (2018).

[11] D. Torre, G. Procaccianti, D. Fucci, S. Lutovac, G. Scanniello, On the presence of green and sustainable software engineering in higher education curricula, in: Proceedings of the 1st International Workshop on Software Engineering Curricula for Millennials, SECM '17, IEEE Press, 2017, p. 54–60.
URL https://doi.org/10.1109/SECM.2017.4

[12] M. Dimitrov, C. Strickland, S.-W. Kim, K. Kumar, K. Doshi, Intel® power governor, https://software.intel.com/en-us/articles/intel-power-governor, accessed: 2017-10-12 (2015).

[13] NVIDIA Management Library, https://developer.nvidia.com/nvidia-management-library-nvml (2021).

[14] M. A. Hoque, M. Siekkinen, K. N. Khan, Y. Xiao, S. Tarkoma, Modeling, profiling, and debugging the energy consumption of mobile devices, ACM Comput. Surv. 48 (3) (2015) 39:1–39:40.
URL http://doi.acm.org/10.1145/2840723

[15] D. Di Nucci, F. Palomba, A. Prota, A. Panichella, A. Zaidman, A. De Lucia, Software-based energy profiling of android apps: Simple, efficient and reliable?, in: 2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER), IEEE, 2017, pp. 103–114.
URL https://doi.org/10.1109/SANER.2017.7884613

[16] Monsoon, Monsoon solutions, inc., http://www.msoon.com/LabEquipment/PowerMonitor/ (2018).

[17] Green Software Laboratory.
URL http://greenlab.di.uminho.pt/

[18] J. Saraiva, R. Abreu, J. Cunha, J. P. Fernandes, GreenSoftwareLab: Towards an engineering discipline for green software, Impact 2018 (1) (March 2018).
URL https://doi.org/10.21820/23987073.2018.9

[19] D. Maia, M. Couto, J. Saraiva, R. Pereira, E-Debitum: Managing Software Energy Debt, in: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering Workshops, ASE '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 170–177.
URL https://doi.org/10.1145/3417113.3422999

[20] R. Rua, M. Couto, J. a. Saraiva, GreenSource: A large-scale collection of android code, tests and energy metrics, in: Proceedings of the 16th International Conference on Mining Software Repositories, MSR '19, IEEE Press, 2019, p. 176–180.
URL https://doi.org/10.1109/MSR.2019.00035

[21] J. Saraiva, R. Rua, Energy Debt: Foundations, techniques and tools, in: SusTrainable Summer School 2022, Rijeka, Croacia, July, 2022, Revised Selected Papers, Springer International Publishing, (to appear).