2024 32nd International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE) | 979-8-3315-0587-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/WETICE64632.2024.00024

RADIANCE: A CASE Tool For Green Software Design

1st Jorge Andrés Larracoechea
LIUPPA
Université de Pau et des Pays de L'Adour
Anglet, France
jorge-andres.larracoechea@etud.univ-pau.fr

2nd Sergio Ilarri
I3A
Universidad de Zaragoza
Zaragoza, Spain
silarri@unizar.es

3rd Philippe Roose
LIUPPA
Université de Pau et des Pays de L'Adour
Anglet, France
Philippe.Roose@iutbayonne.univ-pau.fr

***Abstract*—Regardless of the improvements in the efficiency of energy consumption of information and communication technology, energy consumption will forever be a requisite for software execution. Consequently, researchers have promoted the development of green and sustainable software with new development methods and tools. These, however, have been adopted with limited success due to technicalities and specific language/platform requirements. In this paper, we present RADIANCE: a web app for designing greener software with a model-driven approach based on the Behavior-Based Consumption Profiles (BBCP) external Domain-Specific Language (DSL). RADIANCE, in contrast to other tools, embraces users with different levels of knowledge about green software and software architecture. Moreover, RADIANCE assesses and rates, reports, and provides advice on the energy-consuming patterns of the software models created by the user, assisting them in identifying possible design changes that result in greener software designs from the initial stages of software development.**

***Index Terms*—Software design, Green software, Sustainable software, Energy consumption, Software development, Model Driven Software Development**

I. Introduction

The ongoing energy and environmental crisis has fostered new solutions and approaches to reduce the electrical energy expenses worldwide. Green software (also called sustainable software), has regained the attention of researchers as a contribution to reducing the carbon emissions of Information and Communication Technologies, the culprit for up to 10% of the electrical energy expense worldwide [1]. Green software, as specified by H. Acar [2], can be divided into 3 different categories: green with software, green within the software, and green software. *Green with software* is defined as the use of software to generate frugal solutions to outside problems. *Green within the software* is the use of power models inside the software to make its execution more frugal. Finally, *green software* contemplates the variables during the software development and its surrounding circumstances to save energy throughout the creation of software and results with frugal software.

As previous research has concluded, most of the existing tools and methods available target both the creation of green with software and green within software applications [3]. By demonstratively placing the existing tools on a software development framework such as the traditional waterfall Software Development Life Cycle (SDLC) for comparison, it stands out that all of them target the latter stages (coding, deployment and maintenance), leaving the initial stages (analysis and design) completely unsupported.

In addition, previous studies have found that university students already associate electrical energy expenditure with software execution. Still, they lack the tools and knowledge required to learn and apply energy-saving patterns to the software they learn to design and develop. [4]. Moreover, experienced software developers in the industry identify energy efficiency as a parameter for creating successful software. Nevertheless, stakeholders usually neglect the expense of the overhead time required to trim the energy expense of software [5].

In this paper, we present our tool RADIANCE (softwaRe behAvior DesIgn And eNergy Consumption assEssment). RADIANCE's goals are the following: (1) raise the awareness of its users on the energy-consuming behavioral patterns in software models from the analysis and design stages of a software development project, (2) rate the user's software models according to the presence or lack of energy-consuming patterns, (3) produce agent simulations that allow the users of RADIANCE to understand how their models will behave in mass over time.

The remainder of this paper is structured as follows: section Section II extends the motivations of this paper providing an overview of the available green-oriented development tools, their target stage in software de-

979-8-3315-0587-5/24/$31.00 ©2024 IEEE
DOI 10.1109/WETICE64632.2024.00024

Authorized licensed use limited to: University of L'Aquila. Downloaded on June 12,2025 at 15:30:43 UTC from IEEE Xplore. Restrictions apply.

velopment, and their features compared against RADIANCE. Section Section III presents the architecture of RADIANCE, and Section IV through Section VI present the core features of RADIANCE that difference it from other approaches, to conclude in Section VIII with our future work and research directions, in addition to the closing paragraphs.

II. Related Work

According to the Green Software Foundation, 2,000 specialized green software tools currently exist in open-source, academic, and commercial contexts [6]. As we previously noticed a lack of green software approaches that target the analysis and design phases [3], we decided to extend our previous search for green-oriented software tools. We define green-oriented tools as software tools whose mission or objective is, regardless of the stage of software development they target, to reduce, report, or measure the prospective energy consumption of software or its collateral effects, such as carbon emissions. We compared the tools using the criteria available in Table I, which are a series of needs we identified during the development of a unique DSL we developed for the descripton of software behavior. Additionally, we took note of what stage of the SDLC each tool intended to support or is applicable for.

TABLE I
Criteria used to compare existing approaches to fit our needs

Criteria	Description
C1	The approach allows its users to model the behavior of software.
C2	The approach allows its users to define a change in the behavior of a software's model over time at different levels of granularity.
C3	The approach supports the analysis of software at different levels of granularity.
C4	The approach is oriented towards generating useful metrics for evaluating the energy consumption of software or software models.
C5	The approach includes the means for profiling/assessing human-provided input or services or any way of assessing the impact/unpredictability of human interaction.
C6	The approach is hardware and software agnostic
C7	The approach targets the initial stages of the SDLC or initial stages of a software development project.
C8	The approach is aimed at users with different skill levels to promote the democratization of green software.
C9	The approach exists within a green software method or intends to become part of one.
C10	The approach provides an algorithmic score or rank to the software or software model it analyzes based on its energy consumption, behavior, or other related metrics such as carbon emissions.

As the excerpt of the green-oriented tools we found shows, available in Table II, most of the existing tools and approaches target the development [7]–[23], the deployment [13]–[19], [24], [25] and the maintenance stages [7]–[10], [12]–[25]. None of the tools we studied targeted the analysis and design stages, verifying what we concluded in a previous contribution [3]. Due to this severe gap in tools supporting the analysis and design stages, we developed RADIANCE. A comparison of the features included in RADIANCE against the tools we previously studied is available in Table III.

TABLE II
Distribution of the green-oriented tools we studied along a "classical" waterfall SDLC

Contribution	Analysis & Design	Development	Deployment	Maintenance
AWS Customer Carbon Footprint [24]	×	×	✓	✓
Carat [23]	×	✓	×	✓
CloudCarbon [12]	×	×	✓	✓
CodeCarbon [17]–[19]	×	✓	✓	✓
EcoCode [7]	×	✓	×	✓
EcoGrader [8]	×	✓	×	✓
GCP carbon footprint [11]	×	×	✓	✓
Green Advisor [21]	×	✓	×	✓
Green Frame Model [13]	×	×	✓	✓
Green Miner [20]	×	✓	×	✓
JoularJX [16]	×	✓	✓	✓
Kepler [14]	×	×	✓	✓
PowerJoular [15]	×	✓	✓	✓
SEED [22]	×	✓	×	✓
WebsiteCarbon [9], [10]	×	✓	×	✓
RADIANCE	✓	✓	✓	✓

The Behavior-Based Consumption Profiles (BBCP) external DSL is a software behavior modeling language we developed to describe the evolution of software behavior over time [26], [27]. Its advantages over other software behavior modeling languages are (1) its object-oriented implementation of software modeling that uses concepts familiar to programmers and software architects with any level of expertise, (2) its support for time-based changes in the behavior of software, (3) its ability to include probability in the behavior of software and, finally, (4) the incorporation of hardware consumption in the models created with it. It is used as the basis for software modeling inside of RADIANCE, as the UI and the core features of RADIANCE accommodate the concepts introduced by the BBCP. RADIANCE is also a candidate to be implemented in the design stage of a Full-Stack Green Software Development (GSD) Methodology we previously proposed [28].

Authorized licensed use limited to: University of L'Aquila. Downloaded on June 12,2025 at 15:30:43 UTC from IEEE Xplore. Restrictions apply.

TABLE III
Comparison table of the features available in RADIANCE against other green-oriented tools and approaches

Contribution	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
AWS Customer Carbon Footprint [24]	×	×	✓	✓	×	×	×	×	×	×
Carat [23]	×	×	×	✓	×	×	×	×	×	✓
CloudCarbon [12]	×	×	✓	✓	×	×	×	×	×	×
CodeCarbon [17]–[19]	×	×	✓	✓	×	×	×	×	×	✓
EcoCode [7]	×	×	✓	✓	×	×	×	×	×	✓
EcoGrader [8]	×	×	×	✓	×	×	×	×	×	✓
GCP carbon footprint [11]	×	×	✓	✓	×	×	×	×	×	×
Green Advisor [21]	×	×	✓	✓	×	×	×	×	×	×
Green Frame Model [13]	×	×	✓	✓	×	✓	×	×	×	×
Green Miner [20]	×	×	×	✓	×	×	×	×	×	×
JoularJX [16]	×	×	✓	✓	×	×	×	×	×	×
Kepler [14]	×	×	✓	✓	×	✓	×	×	×	×
PowerJoular [15]	×	×	✓	✓	×	×	×	×	×	×
SEED [22]	×	×	✓	✓	×	×	×	×	×	×
WebsiteCarbon [9], [10]	×	×	×	✓	×	×	×	×	×	✓
RADIANCE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

III. The Architecture

RADIANCE is a web app with no server-side, written entirely in JavaScript and HTML5. Bootstrap, supports the UI among other open-source libraries that make plotting, guiding, and alerting the user easier. JavaScript was selected due to its popularity, ease of use, and pervasiveness on the web. As RADIANCE uses a single-page architecture, all the web components are loaded from the beginning, and the page does not need to reload. Moreover, the distribution of software updates is performed instantly and no installation is needed.

Four main tiers constitute the single-page architecture of RADIANCE: the library, the model builder, the component builder, and the behavioral load consumption simulator. The top tiers depend on the functionality and products of the lower tiers for RADIANCE to aggregate to its full scope in functionality, as seen in Figure 1. The library is the landing page of RADIANCE and, as its name implies, is responsible for managing the software models created by the user. The general purpose of the model builder is to allow its users to swiftly model the behavior of software and acquire an energy rating for finished models. The component builder allows the user to aggregate the models built with the software builder into logical components that can be later rated according to the individual models that compose them. Finally, the products of the previous components of RADIANCE are employed in the behavioral load consumption simulator to execute agent simulations to preview the resource-consuming behavior of the software components created by the user.

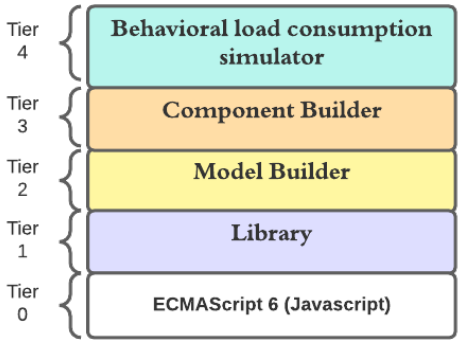

Fig. 1. The tiers that constitute the architecture of RADIANCE

IV. Software Behavior Modeling

The software behavior modeling feature of RADIANCE allows users to quickly create models of the behavior of software using common concepts in software architecture and Object-Oriented Programming (OOP) such as functions, I/O parameters, function sequencing, triggers, and time. The *Model Builder* (tier 2) of RADIANCE, is responsible for facilitating the software behavior modeling process by adapting the available modeling concepts of the BBCP to the user's expertise using UI language levels. UI language levels are reconfigurations of the UI style and natural language that occur when the user selects its level of expertise.

The level of expertise depends on (1) the acquaintance of the user with RADIANCE, (2) the acquaintance of the user with our BBCP DSL, and (3) the familiarity of the user with the description of the consumption of hardware resources (CPU, RAM, etc.). The modeler's UI level can be changed at any time, as RADIANCE supports forward and backward compatibility between UI's language configurations, meaning that the user can choose to model on whichever level of expertise they prefer on the fly, covering the need for a tool that is easily introduced and used by beginners. UI levels can also be customized to improve the presentation or the user experience.

The model builder is also capable of executing an algorithm that classifies the energy-consuming patterns of each function similarly to the ones in Figure 2, and generates a step-by-step classified timing diagram of the sequential and parallel function calls similar to the sample in Figure 3. The algorithm used by RADIANCE can be customized according to the needs of the users, and a custom algorithm created by us will be included

Authorized licensed use limited to: University of L'Aquila. Downloaded on June 12,2025 at 15:30:43 UTC from IEEE Xplore. Restrictions apply.

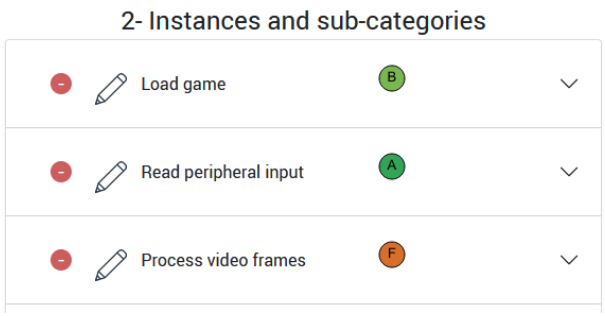

Fig. 2. Three functions labeled by their energy-consuming behavioral patterns

in the latest release. The classification generated by the current algorithm depends on the value that the users select for each property available in the model, and its underlying mechanics in the BBCP. Each function and step are labeled with a letter and a color from A in green (most frugal) to G in red (most energy-consuming), similar to the classification of the EU's energy label. After the classification of functions and function calls, the user is presented with a prospective energy consumption report similar to the sample in Figure 4, that (1) warns the user of the possible high energy consuming behaviors in the model, (2) provides advice on good practices to conserve low energy consumption and, (3) fosters energy-saving behavioral patterns with positive feedback.

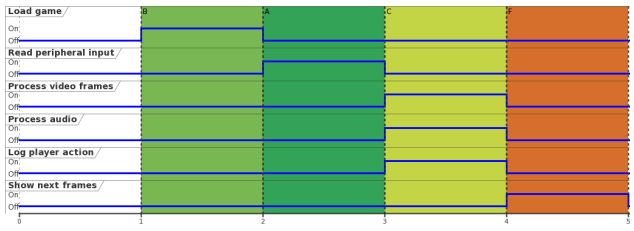

Fig. 3. A timing diagram created, classified, and labeled of a software model using RADIANCE

V. Component Behavior Modeling

As mentioned in Section IV, the model sequencer allows users to create cross-model interactions and dependencies. To allow the users to model more complex applications or software components, we included a component behavior modeler in the component builder (tier 3 of Figure 1). To control how models can be grouped, the component modeler uses "collections": logical agglomeration of individual models or nested collections. Models in the same collection can share parameters and triggers (logical conditions for setting a model in an active state). Inheritance among children

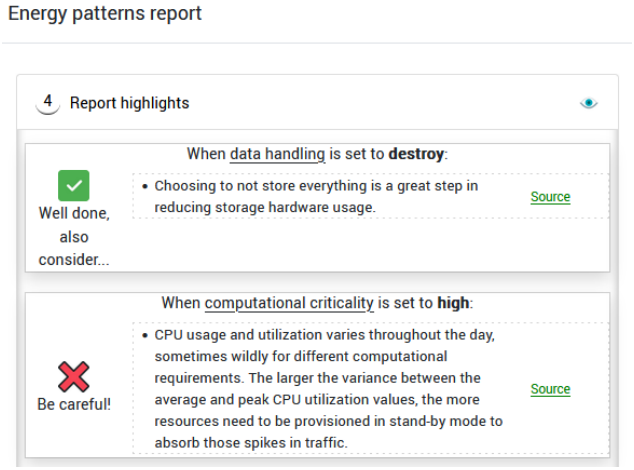

Fig. 4. A snapshot of an energy patterns report created with RADIANCE

collections is also supported. Similarly to the behavior modeler, the component behavior modeler is capable of executing a collection rating algorithm. We will soon propose an algorithm to rate collections so that the user can easily comprehend the impact on the energy/hardware consumption of grouping specific models and collections.

VI. Behavioral Load Simulator

Taking advantage of the previous features in RADIANCE and in addition to the characteristics that make the BBCP different from other software behavior description languages, we embedded a behavioral load simulator in RADIANCE. The behavioral load simulator uses an engine tailored to interpret the variables of BBCP-supported models, interpreting models into agents. Collections can be configured as logical spaces where the member agents' consumption is mitigated, data policies are enforced, and the projected energy consumption can be increased or decreased to account for diverse scenarios such as power source or hardware environments. As BBCP models support the re-definition of certain variables during specific timeframes in a simulation, the behavior of models can change through time. Furthermore, if a user would like to include uncertainty as part of a model's behavior, it is possible to define specific probabilities for an agent to enter an active state or an inactive state. The probabilities can change at specific time frames at predefined rates.

The results of the engine, once it has been configured to interpret one or more collections, are a dataset of the usage created by each agent and a plot graph showing the change in state for each agent. A sample

Authorized licensed use limited to: University of L'Aquila. Downloaded on June 12,2025 at 15:30:43 UTC from IEEE Xplore. Restrictions apply.

of the plot graph is available in Figure 5, showing the results in the change of the binary state (active and inactive) of 3 different models throughout 2 simulated hours.

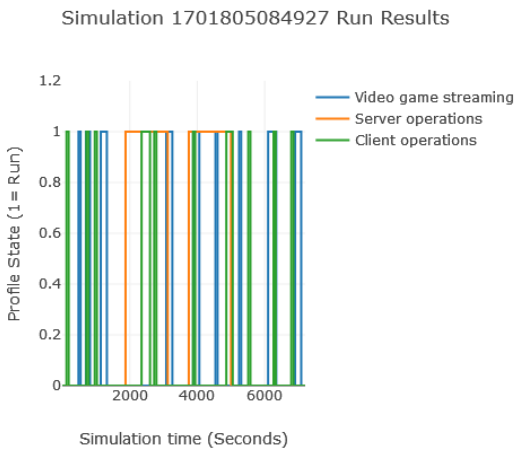

Fig. 5. A sample of the change in the state of 3 profiles between active and inactive throughout 2 simulated hours

The data generated by the behavioral load simulator can be later used by software deployment simulators so that, during the design stage, the users of RADIANCE can understand the future challenges that will arise during the development and deployment phases, and anticipate them proactively.

VII. Incorporating RADIANCE Into a GSD Method

Our proposal to incorporate RADIANCE into the multiple stages of a GSD method is the following:

1) Introduce RADIANCE to software designers and architects in the design stages of the project, resulting in:
   a) A description of behavior, expected hardware consumption and functional dependencies, at the individual function level.
   b) Recommended practices to mitigate energy consumption, the parallel or sequential flow of function calls, and an expected share of hardware consumption per function, at the single model level.
   c) The dependencies management among components and an expectation of component hardware consumption, at the component level.

   Which are later passed on to the DevOps (development and operations) team.
2) The DevOps team uses the results of the design stage to:
   a) Identify and implement the best practices in the code and technology stack, in addition to other tools such as SEED [22] or EC0lint for the code [29], and JoularJX [16] for testing the prototype's consumption.
   b) Swiftly re-adjust the models in RADIANCE to categorize the actual hardware consumption of the prototype.
   c) Employ the Behavioral Load Simulator to generate the estimated prospective usage data based on the adjusted software models.
3) The DevOps team uses the prospective usage data of the models to:
   a) Create proactive orchestration algorithms that account for the evolution of hardware usage throughout time using re-deployment simulators like PISCO [30].
   b) Obtain information about the prospective consumption of their stack at the deployment level with tools such as Cloud Carbon [31] or Green Cost Explorer [32].
   c) Re-adjust the technology stack for the final deployment.

Any of the previous steps proposed can be revisited to accommodate new changes that contribute to more frugal software.

VIII. Conclusion and Future Work

Throughout the paper, we presented RADIANCE: a tool for designing models of the behavior of software using model-driven design, and assessing and cataloging the energy-consuming patterns of software models. Moreover, we introduced the objectives, components, features, and its general architecture. Thanks to its flexibility, RADIANCE will be a stepping stone toward the democratization of green software by facilitating green software design at diverse levels of complexity and granularity. A limited build of RADIANCE is available in the following URL: https://joalago.github.io/RADIANCE-demo/. We are currently integrating RADIANCE with PISCO [30], a simulator for microservices deployment. This integration will allow us to replace the static hardware resource consumption simulated by PISCO with a dynamic hardware consumption backed by the software models and behavior generated with RADIANCE, to study and create more efficient energy-saving algorithms for the deployment stage of the SDLC.

As a future work, we will perform a usability test on RADIANCE to prove 3 major points: that the current UI expertise levels are useful to the public they are targeted at, and that the report and the energy category

Authorized licensed use limited to: University of L'Aquila. Downloaded on June 12,2025 at 15:30:43 UTC from IEEE Xplore. Restrictions apply.

affect the decision-making of the users, contributing to the latter stages of the SDLC.

Acknowledgment

This publication is part of the project PID2020-113037RB-I00, funded by MICIU/AEI/10.13039/501100011033. Besides the NEAT-AMBIENCE project, we also thank the support of the Departamento de Ciencia, Universidad y Sociedad del Conocimiento del Gobierno de Aragón (Government of Aragon: Group Reference T64_23R, COSMOS research group).

References

[1] E. Gelenbe, "Electricity Consumption by ICT: Facts, trends, and measurements," *Ubiquity*, vol. 2023, no. August, pp. 1:1–1:15, Aug. 2023.

[2] H. Acar, "Software development methodology in a Green IT environment," Ph.D. dissertation, Université de Lyon, Nov. 2017.

[3] J. Larracoechea, P. Roose, S. Ilarri, Y. Cardinale, S. Laborie, and M. González, "Towards services profiling for energy management in service-oriented architectures," in *17th International Conference on Web Information Systems and Technologies*, Jan. 2022, pp. 209–216.

[4] C. Pang, A. Hindle, B. Adams, and A. E. Hassan, "What Do Programmers Know about Software Energy Consumption?" *IEEE Software*, vol. 33, pp. 1–1, Jan. 2015.

[5] E. Jagroep, J. Broekman, J. M. E. M. van der Werf, S. Brinkkemper, P. Lago, L. Blom, and R. van Vliet, "Awakening awareness on energy consumption in software engineering," in *Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Society Track*, ser. ICSE-SEIS '17. Buenos Aires, Argentina: IEEE Press, May 2017, pp. 76–85. [Online]. Available: https://doi.org/10.1109/ICSE-SEIS.2017.10

[6] "There are 2,000 specialized green software tools available in open-source, commercial, and academic contexts | State of Green Software," https://stateof.greensoftware.foundation/insights/green-software-specialized-tools/.

[7] "GitHub - green-code-initiative/ecoCode: Reduce the environmental footprint of your software programs with SonarQube," https://github.com/green-code-initiative/ecoCode, accessed on 06-05-2024.

[8] "Ecograder," https://ecograder.com/how-it-works, accessed on 06-05-2024.

[9] "How does it work?" https://www.websitecarbon.com/how-does-it-work/, accessed on 06-05-2024.

[10] "Introducing the Website Carbon Rating System," https://www.websitecarbon.com/introducing-the-website-carbon-rating-system/, accessed on 06-05-2024.

[11] "Carbon Footprint reporting methodology | Carbon Footprint Documentation," https://cloud.google.com/carbon-footprint/docs/methodology, accessed on 06-05-2024.

[12] "Methodology | Cloud Carbon Footprint," https://cloud-carbon-footprint.github.io/docs/methodology, accessed on 06-05-2024.

[13] "Greenframe-cli," https://github.com/marmelab/greenframe-cli/blob/main/src/model/README.md, accessed on 06-05-2024.

[14] M. Amaral, H. Chen, T. Chiba, R. Nakazawa, S. Choochotkaew, E. K. Lee, and T. Eilam, "Kepler: a framework to calculate the energy consumption of containerized applications," in *2023 IEEE 16th International Conference on Cloud Computing (CLOUD)*, Jul. 2023, pp. 69–71.

[15] "PowerJoular," accessed on 06-05-2024. [Online]. Available: https://www.noureddine.org/research/joular/powerjoular

[16] A. Noureddine, "PowerJoular and JoularJX: multi-platform software power monitoring tools," in *2022 18th International Conference on Intelligent Environments (IE)*, Jun. 2022, pp. 1–4.

[17] "CodeCarbon.io," https://codecarbon.io/, accessed on 06-05-2024.

[18] "Mlco2/codecarbon," https://ec0lint.com/get-started, Jan. 2024, accessed on 06-05-2024.

[19] "Visualize — CodeCarbon 2.3.4 documentation," https://mlco2.github.io/codecarbon/visualize.html, accessed on 06-05-2024.

[20] A. Hindle, A. Wilson, K. Rasmussen, E. J. Barlow, J. C. Campbell, and S. Romansky, "GreenMiner: a hardware based mining software repositories software energy consumption framework," in *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*. Hyderabad, India: ACM Press, 2014, pp. 12–21.

[21] K. Aggarwal, A. Hindle, and E. Stroulia, "GreenAdvisor: a tool for analyzing the impact of software evolution on energy consumption," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2015, pp. 311–320.

[22] I. Manotas, L. Pollock, and J. Clause, "SEEDS: a software engineer's energy-optimization decision support framework," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: Association for Computing Machinery, May 2014, pp. 503–514.

[23] A. J. Oliner, A. P. Iyer, I. Stoica, E. Lagerspetz, and S. Tarkoma, "Carat: collaborative energy diagnosis for mobile devices," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '13. New York, NY, USA: Association for Computing Machinery, Nov. 2013, pp. 1–14.

[24] "Carbon Footprint Reporting – Customer Carbon Footprint Tool – Amazon Web Services," https://aws.amazon.com/aws-cost-management/aws-customer-carbon-footprint-tool/, accessed on 06-05-2024.

[25] "Calculating My Carbon Footprint | Microsoft Sustainability," https://www.microsoft.com/en-us/sustainability/emissions-impact-dashboard.

[26] J. A. Larracoechea, S. Ilarri, and P. Roose, "A proposal of behavior-based consumption profiles for green software design," *Applied Sciences*, vol. 14, no. 17, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/17/7456

[27] J. Larracoechea, P. Roose, S. Ilarri, Y. Cardinale, S. Laborie, and O. Vara, "Behavior-based consumption profiles for the approximation of the energy consumption of services," *International Conference on Information Systems Development (ISD)*, Sep. 2022. [Online]. Available: https://aisel.aisnet.org/isd2014/proceedings2022/usability/1/

[28] P. Roose, I. Sergio, J. A. Larracoechea, Y. Cardinale, and S. Laborie, "Towards an integrated full-stack green software development methodology," in *29th International Conference on Information Systems Development*, Sep. 2021.

[29] "Ec0lint," https://ec0lint.com/get-started.

[30] H. Humberto Alvarez Valera, M. Dalmau, P. Roose, J. Larracoechea, and C. Herzog, "PISCO: a smart simulator to deploy energy saving methods in microservices based networks," in *2022 18th International Conference on Intelligent Environments (IE)*, Jun. 2022, pp. 1–4.

[31] "Overview | Cloud Carbon Footprint," https://cloud-carbon-footprint.github.io/docs/, accessed on 06-05-2024.

[32] "The green web foundation/green-cost-explorer," The Green Web Foundation, Dec. 2023, accessed on 06-05-2024.

Authorized licensed use limited to: University of L'Aquila. Downloaded on June 12,2025 at 15:30:43 UTC from IEEE Xplore. Restrictions apply.