



HAL
open science

Uncovering the Environmental Impact of Software Life Cycle

Thibault Simon, Pierre Rust, Romain Rouvoy, Joël Penhoat

► **To cite this version:**

Thibault Simon, Pierre Rust, Romain Rouvoy, Joël Penhoat. Uncovering the Environmental Impact of Software Life Cycle. International Conference on Information and Communications Technology for Sustainability, Jun 2023, Rennes, France. hal-04082263

HAL Id: hal-04082263


<https://inria.hal.science/hal-04082263>

Submitted on 26 Apr 2023


HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.


L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Uncovering the Environmental Impact of Software Life Cycle

Thibault Simon 
Orange Labs, Inria
Rennes, France
thibault.simon@inria.fr

Pierre Rust 
Orange Labs
Rennes, France
pierre.rust@orange.com

Romain Rouvoy 
Univ. Lille, Inria, CNRS,
UMR 9189 CRISTAL, France
romain.rouvoy@univ-lille.fr

Joël Penhoat 
Orange Labs
Rennes, France
joel.penhoat@orange.com

Abstract—The environmental footprint of a software service development may be impacted by several factors, spanning human factors to infrastructure choices. To the best of our knowledge, the state of the art in the domain of environmental footprint fails i) to address the full spectrum of software services life cycle, from design to operation and maintenance and ii) to cover different categories of impacts. In this paper, we therefore introduce a methodology and the associated model to guide the software development ecosystem and their stakeholders in estimating the environmental footprint of their projects along multiple impact categories. This holistic approach delivers actionable insights to identify potential shifting between phases, such as development and usage, but also hotspots among resources consumed to produce and operate software services. We report on a sample case study that emphasizes the importance of development impact on a software life cycle and shows the relative importance of phases, as well as resources consumed.

Index Terms—software life cycle, software environmental impact, green software development

I. INTRODUCTION AND MOTIVATION

The latest *Intergovernmental Panel on Climate Change* (IPCC) report [31] states that emissions of greenhouse gas from human activities are responsible for approximately 1.1°C of global warming since 1850–1900 and that—unless there is an immediate, rapid, and large-scale reduction—limiting it to +1.5°C will be out of reach. To achieve this objective, the ICT sector is challenged to adopt a more sustainable approach to reduce its environmental footprint. Freitag *et al.* [21] estimate, based on previous studies [4, 7, 45], that it contributes between 2.1% and 3.9% to worldwide *Greenhouse Gas* (GHG) emissions and continues to grow, while it should reduce by 42% by 2030, 72% by 2040, and 91% by 2050 and be net-zero by 2050 to stay within 1.5°C warming. According to Masanet *et al.* [46], despite an increase of compute instance by 550% between 2010 and 2018, the consumption was stable from 194 to 205 TWh, thanks to the progress of servers and data centers efficiency. However, it is uncertain whether future optimizations will be able to keep up with the rise of computing resource demand.

The most commonly used category of impact is *climate change*, quantified through GHG, but according to best practices and standards [36, 34, 35] several should be combined to get a global environmental footprint. *Material footprint* refers to the total amount of raw materials extracted (minerals, metals...), notably for ICT hardware manufacturing, and reflects

the pressures placed on the environment. Overall, it rises from 43 billion metric tons in 1990 to 92 billion in 2017, an increase of 70% since 2000, and 113% since 1990 [17]. The ICT sector also consumes water, and Siddik *et al.* [63] estimate the total annual operational water footprint of US data centers used for cooling and energy generation at 513,000,000 m^3 in 2018. At the end of its life, ICT hardware becomes e-waste, which is estimated to be recycled at less than 40% in the European Union [66]. Its disposal generates disparities around the world as well as irremediable pollution to soil [47].

Although the ICT environmental impact comes from hardware, especially its manufacturing and usage, one can argue that the responsibility lies within the software. Indeed, ICT equipment are built and operated solely to run software on, therefore levers for actions must also be sought at the software level.

When developing software, developers follow documentation and guidelines to optimize their maintainability, performance, and usability. However, they lack guidance and knowledge on how to lower their environmental impact [51], as well as sustainability models [39, 44, 16, 50, 61]. They are however prone to apply them, if available and easily integrable [65].

To the best of our knowledge, current efforts are mainly focused on lowering software energy footprint by applying performance optimizations [41], such as energy-efficient virtual machines placement in data-centers [48], energy monitoring along releases [37], or energy code-smells [11]. However, the energy consumed throughout the usage phase of a service or product might not always be the main driver of impact, thus adopting a more holistic approach is essential. For instance, Gupta *et al.* [26] showed that reducing a data center’s energy consumption alone fails to reduce its carbon emissions, as hardware manufacturing plays a bigger role in its environmental impact. Therefore, to account for these effects in software, we need to model its whole life cycle, across all its phases and over multiple environmental criteria.

Our study aims at answering the following research question: *How does a holistic perspective provide actionable insights to reduce software environmental footprint?* . To do so, we propose a model to ease the estimation of software environmental impact through its life cycle across multiple categories, to study potential shifts and hotspots.

The remainder of this paper is organized as follows. We start by describing the key standards on life cycle analysis and software life cycle (cf. Section II). We then discuss the state of the art on leveraging software life cycle to lower its environmental impact (cf. Section III). We later introduce our contribution to model a software life cycle and estimate its environmental impact (cf. Section IV). To foster widespread adoption of our approach, we propose an implementation in Section V. We subsequently show how it can help stakeholders to identify impact shifting between phases, resources, and criteria (cf. Section VI). Finally, we conclude in Section VIII.

II. BACKGROUND

This section describes the key standards on life cycle analysis and software life cycle. In our work, we rely on these standards, simplifying and complementing them when necessary to provide a holistic view of the environmental impact.

A *Life Cycle Assessment (LCA)*, as defined by the standard ISO 14040 [34], is the study of the environmental impacts contribution of a product or service across its entire life cycle. The analysis is performed for a given *functional unit*, allowing to compare two systems with the same one from an environmental perspective. By using a systematic overview and perspective, the LCA approach helps to identify the shifting of a potential environmental burden between life cycle stages or individual processes [19]. A *screening LCA* is simpler to conduct and is established based on readily available data [67], and thus provides a high-level view of impacts to identify the main sources and those that require deeper examination.

Ekvall [18] defines two types of LCA: *attributional*, which aims at describing the environmental properties of a life cycle and its subsystems, and *consequential* which aims at describing the effect of changes within the life cycle. Whereas a consequential LCA aims at describing the long-term effect of changes induced by the product or service and thus support decisions in the long term, an attributional one maps the environmental impacts that a product can be made accountable for and is thus better suited to support decisions that aim at improving its life cycle processes [59].

Guinée *et al.* [27] define an environmental impact as the consequence of an environmental intervention in the environment system. It is represented as an aggregate of impact categories, a class representing an environmental issue of concern [34], and groups consequences on climate change, acidification, ecotoxicity... Each one is represented as a quantity, which unit is an impact category indicator. For example, the impact category *Climate Change Potential* contains a quantity whose unit is the *tons of Carbon Dioxide Equivalent (tCO₂e)*, defined as the mass of CO₂ that would yield an equivalent effect of global warming [14].

To define a *Software Development Life Cycle (SDLC)*, the international standard ISO/IEC/IEEE 12207 [33], entitled *Systems and software engineering - Software life cycle processes*, defines a set of processes organized as requirements, design, implementation, testing, deployment, maintenance and

retirement [62]. In projects, these life cycle processes can further be organized in different models, such as Waterfall, V-Model, Scrum... [57]

III. STATE OF THE ART

The trends in green computing studies published since 2012 clearly show a growing interest [13], and several approaches [16, 40, 42, 44, 50, 62] attempt to leverage software life cycle models to reduce its environmental impact.

Some authors propose to use the life cycle of software to lower the impacts of the different processes. Kumar *et al.* [40] propose the Green Star model to assess and rate each process of the SDLC. Lami *et al.* [42] look at the sustainability factors of software processes and propose the following reduction objectives: carbon footprint, energy, waste, and travel. They also add new processes to assess that these objectives are fulfilled.

Processes can also be used to reduce the resulting software's environmental impact. Shenoy *et al.* [62] propose changes in the SDLC and provide guidelines for each process. Dick and Naumann [16] add processes dedicated to ensuring the resulting product sustainability, such as sustainability reviews and sustainability retrospectives. The GREENSOFT model and associated metrics [50, 44] cover both the process and resulting software sustainability within a conceptual framework that emphasizes considering software products over their life cycle.

The approaches for computing an environmental impact can be classified as *top-down* and *bottom-up*. A top-down approach adopts global statistics and input/output analysis to model the participating systems and compute a smaller-scale output. Chan *et al.* [12] use this approach to model the Internet energy intensity, by dividing its total estimated energy consumption by the traffic on a given period. They obtain an intensity expressed in *kWh-hours per gigabyte of data transferred*. Unfortunately, top-down approaches generally suffer from an averaging bias and do not help in identifying action levers. In this case for example, one can only identify two levers to reduce the Internet's impact: either reducing the amount of data transferred or reducing the environmental footprint of the ICT sector as a whole.

A bottom-up approach is based on direct observations made on case studies [15], which are then generalized to be representative of a larger scale. Schien *et al.* [60] use this approach to compute the Internet energy intensity, by scaling the energy consumption of devices simulating an *Internet Service Provider (ISP)* network. This approach will reflect changes made to the system directly, without relying on structural changes, as a top-down approach.

Most studies on the environmental footprint of ICT combine both approaches to obtain more accurate estimations [21]. Malmodin and Lundén [45] used a bottom-up approach for the impact of user devices, and a top-down approach for data centers and networks to model their projections of emissions beyond 2020.

The lack of openly available data, the complexity of ICT systems and their variability make it difficult to assess their

impact. While Andrae *et al.* [5] estimate the worldwide data centers consumption to be around 299 TWh in 2020, China's State Grid Energy Research Institute released a report stating that Chinese data centers consumption alone was around 200 TWh the same year [24]. Modeling various systems implies adopting hypotheses and values with possibly high margins of errors and, per Fonseca *et al.* [20], one should look at trends that help to understand patterns rather than absolute values to take sourced decisions.

To the best of our knowledge, only two papers tried to assess the carbon footprint of software products' life cycle. The first attempt was conducted by Taina *et al.* [64], where the authors proposed different approaches to assess the carbon footprint of phases on an artificial software project. By giving estimates for each phase, they emphasize that the way software is developed and delivered matters, as well as how usable it is. They mostly focus on energy consumption and paper printing, but conclude that traveling would dominate the result, if included. Kern *et al.* [38] focused on whether employee commuting should be included in software carbon footprint assessment, and conduct two case studies on micro-enterprises in Germany. They compute factors of impact for this context to give an impression of the magnitudes, and show that the distribution of environmental impacts between the build and the run phases can vary greatly depending on the number of copies sold. Both of these approaches do not cover the complete life cycle of the resources used, especially the impact of ICT equipment manufacturing, but only account for their energy consumption.

The ITU L.1410 [36] standard complements the ISO 14040 [34] and 14044 [35] and proposes a *Methodology for environmental life cycle assessments of information and communication technology goods, networks and services*, by defining a set of requirements that the LCA practitioners should strive for, as compliance may not be possible. It was refined in France by the *General Principles for the Environmental Labelling of Consumer Products, Methodological Standard for the Environmental Assessment of Digital Service* [22], which aims to provide a set of common rules to assess and inform consumers about the environmental impact of digital services. The Blue Angel environmental label for *Resource and Energy-Efficient Software Products* is designed in Germany to award products that use hardware resources in a particularly efficient manner and consume a low amount of energy during their use. All these standards take only into account the usage of software services, but not their development.

The *Greenhouse Gas Protocol Product Life Cycle Accounting and Reporting Standard (Product Standard)* [8] gives requirements and guidance for organizations to quantify and report an inventory of GHG emissions and removals associated with a specific product. The *ICT Sector Guidance* [53] define the assessment of software life cycle GHG impact with predefined phases (material acquisition and preprocessing, production, distribution and storage, use, and end of life), and consider the software development. However, unlike the

LCA approaches, it uses a single category of impact and only accounts for GHG emissions. Besides, upstream emissions such as building and hardware manufacturing, are not taken into account.

Outside the research community, several initiatives also propose emerging methodologies. The Green Software Foundation defines the *Software Carbon Intensity* (SCI) [25], a methodology to compute the rate of carbon emissions per functional unit for a software system. It takes into account both energy consumption and reserved hardware embodied emissions, but only covers GHG and the usage phase of a project, not its development. The *Sustainable Digital Infrastructure Alliance* (SDIA) proposes a methodology to compute the carbon emissions of server-side applications [1]. This approach encompasses both energy consumption and embodied carbon emissions, but only accounts for backend impacts and excludes those of the network and end-user devices.

Despite a growing interest to estimate the environmental impact of ICT, few approaches consider software over its life cycle along multiple categories. Standards and indicators emerge to standardize its computation and communication to consumers, but they focus mainly on its usage phase, often only through energy consumption, and its climate change impact through carbon emissions rather than considering multiple categories.

IV. APPROACH

We propose a methodology and the associated model to assist software project stakeholders in estimating the environmental footprint of their project. When designing this methodology, our goals are threefold. It needs to be: 1) *holistic* by providing a comprehensive perception of the impacts resulting from a software project, 2) *actionable* by delivering actionable insights to project stakeholders, 3) *easy to use* to enforce a widespread adoption.

To meet our first objective, we elected to follow the core principles of an attributional screening LCA, as introduced in Section II. The perimeter we take into account in our methodology includes all the resources consumed when performing the activities identified in the SDLC, but also other activities to consider a complete *cradle-to-grave* view of the impacts arising from the ICT project under study. This includes for example (and not exhaustively) the distances traveled to sell the product, the man-days required for its development, the production and deployment of ICT equipment and the impact of end-user devices. This offers a high-level view of the impacts and supports identifying key causes within the processes. Additionally, such a holistic perspective allows the identification of potential burden shifts between life cycle phases or resources.

For a software product, such a shift can occur when reducing the hardware resources required, and thus the running phase impact, by adopting a more efficient programming language. However, this programming language might incur longer development time, which will increase the impacts of

the building phase. Impact shifts can also occur between categories, for instance when replacing servers with more energy-efficient ones, thus decreasing the climate change impact, while increasing the land pollution due to the manufacturing of new hardware.

The adoption of a bottom-up approach allows us to meet our second objective: delivering actionable insights. Once stakeholders have identified major impact sources in their project, they can investigate the effect of decisions taken at a micro-scale on a larger scale. Our approach therefore equips stakeholders with a new *Key Performance Indicator* (KPI) to consider the environmental impacts of their decisions and assess if the benefit outweighs the cost next to others, such as security level, financial costs, redundancy, high availability. . .

Meeting our third objective, namely "*ease of use*", is generally difficult when using a bottom-up approach with a large scope. We answer this challenge by proposing a simple and flexible way of modeling a project's resources and activities, which allows using the same methodology for projects using different development models. Our estimates rely on an impact dataset (named *secondary data* in LCA vocabulary), that we have consolidated from existing openly available sources. Finally, we elect to simplify inventory and data collection by using a static view of the project, which we introduce in the following section.

A. Software life cycle modeling

To adapt to each stakeholder's context, the software life cycle modeling has to be suitable for all types of SLDC models, while keeping a common environmental impact computation methodology.

To do so, it is represented as a static view rather than a time-based one, such as a timeline, to lower the granularity of inputs required while keeping their relevance. For instance, it is not required for stakeholders to detail on each day how many developers worked on the project, the number of test servers used, which would be tedious without automation, but rather the number of *man-days* and *server-hours* consumed on a given period. Instead of a *day-to-day* basis, time is flattened and the life cycle is modeled as a tree, as depicted in Figure 1. In this example, the project contains two key phases, *build* and *run*, which can contain sub-phases, such as *selling*, *development*, and *hosting*. Stakeholders can add, move, and remove phases and sub-phases to model their software development life cycle, no matter the model and processes or phases they chose. To support them do so, we propose predefined trees that model the most common life cycles, which they can fully customize to model more accurately their daily activities.

Each phase consumes resources to be completed, which can be of multiple kinds. The phase *coding* consumes *man-days*, but also *server-hours* for developers to test their products. As software projects use common resources, we propose a default dataset with their associated environmental impact, which the model then uses to compute the whole software life cycle impact. This way, we allow stakeholders to set their

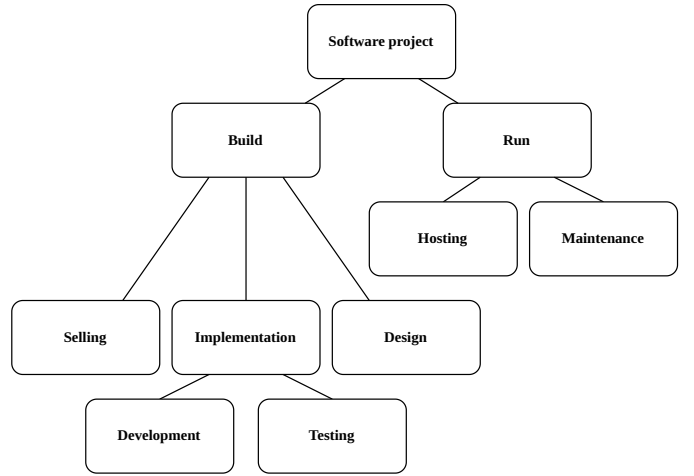


Fig. 1. Software life cycle tree example

resources usage accordingly to their needs and policies, in the right life cycle phases, while keeping the same computation methodology between multiple typologies of projects.

B. Environmental impact computation

As the project life cycle is flattened as a tree representation, its whole environmental impact, represented as an aggregate of impact categories as defined in Section II, can be computed using a tree traversal:

$$Impact(project) = \sum_{p \in P} Impact(phase_p)$$

Each phase of the project—or node of the tree—can consume resources and require sub-phases. For example, the phase *build* requires the sub-phase *development*, which consumes *man-days* resources. As such, the environmental impact of a phase is the sum of its consumed resources, including the ones consumed by its sub-phases:

$$Impact(phase) = \sum_{r \in R} Impact(resource_r) + \sum_{p \in P} Impact(phase_p)$$

The impact of a given phase mostly depends on the resources it consumes. Each one is estimated from the declared quantity weighted to its associated *impact factor*, which gives an environmental impact per functional unit consumed, such as a *vCPU-hour*, a *plane-kilometer*. . . A resource impact is therefore computed as:

$$Impact(resource) = quantity \times impact_factor$$

The quantity has to be the same unit as the impact factor to obtain an environmental impact. However as the model uses a flattened time representation, this quantity cannot directly express recurring events such as a product owner going to meet the customer every month, by plane, for 6 months. The quantity is therefore aggregated from multiple fields: *amount*,

the number of functional units used, *period*, the length of the resource consumption, and *frequency*. For instance, for the impact factor *plane*, expressed by kilometer, the amount will be *100 plane-kilometers*, the frequency *every month*, and the period *a year*, giving a quantity equal to 12,000 *plane-kilometers*, which corresponds to the impact factor functional unit.

If an impact factor is given by a time-based functional unit, such as vCPU-per-hour, the quantity field has to contain a time unit. A *period* of usage is therefore mandatory and the *frequency* field cannot be used in isolation anymore, to avoid canceling out the time unit. The field *duration* is added to represent a recurring scenario: to model a vCPU used two hours per day for a year, the duration will be *a year*, the frequency *every day*, and the duration *two hours*. The quantity value is thus computed as follows:

$$\text{quantity} = \frac{\text{amount} \times \text{duration} \times \text{period}}{\text{frequency}}$$

V. ENVIRONMENTAL MODEL

To foster widespread adoption of our approach to assess software environmental impact, we developed a web application shown in Figure 2 as well as an API implementing our approach, which will soon be available under Orange GitHub ¹. This application provides software project stakeholders with an easy way to model the life cycle of their project and get a first overview of the associated environmental impacts to quickly identify hotspots in their projects. They can model and save their project characteristics, through phases and resources consumed, and analyze the resulting impact from various standpoints.

a) Tree view: Shown at the center of Figure 2, the tree view defines the project life cycle, by modeling its phases as nodes. Each node can be moved or deleted, and new ones can be created at all levels. Stakeholders can adopt a life cycle granularity that suits their study scope, which they can refine iteratively by updating the tree. Standard predefined life cycle trees are proposed to start modeling from the most common SDLC models.

b) Resources inputs: By switching to editing mode, the resources consumed by each phase can be updated. A form assists in conforming with the associated impact factor unit, while allowing entering time-based events despite the overall static view.

c) Impact factors: The impact factors are exposed by the API, and cannot be updated when modeling projects. Keeping the reference values in a single place, rather than distributing them, as spreadsheets for instance, avoid data tempering that would make comparisons between projects or solutions impossible. It also allows updating at once all the projects modeled using these impact factors, when refining their values.

TABLE I
SOFTWARE EFFORT DISTRIBUTION (PERSON-MONTHS)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	368.4	1263.0	3332.9	736.7
Environment/CM	263.1	842.0	1666.5	263.1
Requirements	999.9	1894.5	2666.3	210.5
Design	499.9	3789.0	5332.7	210.5
Implementation	210.5	1368.2	11331.9	999.9
Assessment	210.5	1052.5	7999.0	1263.0
Deployment	78.9	315.7	999.9	1578.7

d) Results: To identify impact hotspots and shifts, the resulting environmental impact for the chosen impact category is displayed on the left side of the application, shown in Figure 2. The top figure is the Sankey diagram of the impact combined between the phases and resources, which is then refined solely by phases, as shown in Figure 3, and by resource, as shown in Figure 4. For the impact by resource, all impact categories are also reported simultaneously, as depicted in Figure 6.

e) Comparison: Each project can contain multiple models, which can be used to compare different approaches and scenarios. Shown on the left-hand side of Figure 2, these models can be compared between them, using a side-by-side view of the environmental impact described previously, as well as a direct comparison by resources shown in Figure 5.

f) Physical quantities: Each physical quantity, such as ones expressed in CO₂e, is represented as a numerical value and a unit of measurement. Our implementation uses the Pint library [28], which allows conversions between units as well as ensures their consistency in computations.

VI. FINDINGS

This section demonstrates the benefits of our approach. We used the Gitlab project to estimate a sample application life cycle, considering the source code repository for modeling the building phase and the documented hardware requirements for the usage phase.

Using the CLOC tool [3], we extracted 6,241,291 *Source Lines of Code* (SLOC) from the project repository [23]. We used this value as an input for a *Constructive Cost Model II* (COCOMO II), the most commonly-used algorithmic method for software development cost estimation defined by Bohem *et al.* [9]. Using a *postmortem* approach on an already developed application offers an estimation of the effort required to build a software, as well as its distribution among phases. To obtain a standard software effort distribution, we leave COCOMO cost drivers, such as team cohesion, programmer’s capability, and multi-site developments with their default values. We obtained the effort distribution shown in Table I.

As discussing the COCOMO potential limits is out of this paper’s scope, we consider the effort distribution as representative of a project this size, rather than the actual effort accomplished to design and develop the Gitlab application.

To model the usage phase, we adopted the Gitlab hosting reference architecture for 50,000 users [54]. We chose the largest one, considering that the application is designed to

¹<https://github.com/Orange-OpenSource>

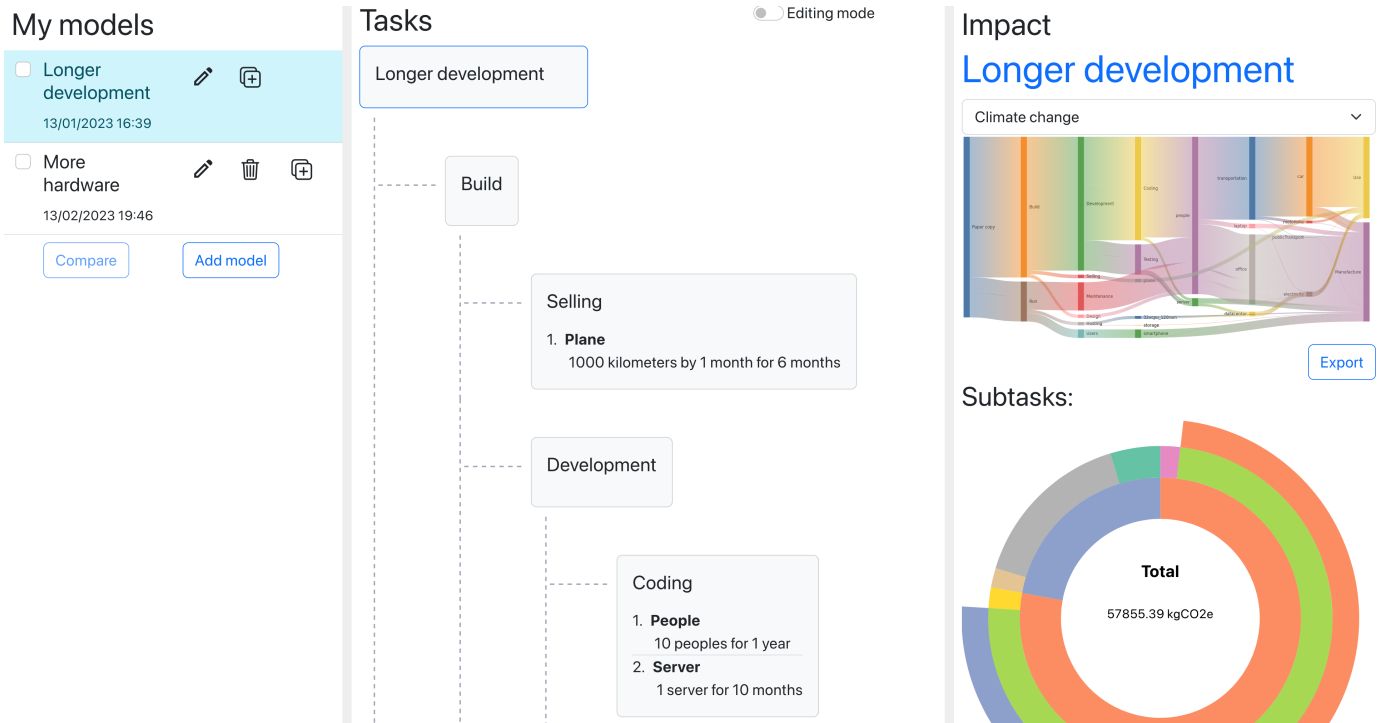


Fig. 2. Screenshot of the web application implementing our approach

be as scalable as its biggest instantiation, requiring more effort than for a smaller one, such as for 500 users. Hosting an instance requires hosting services not developed in-house by Gitlab, such as Sidekiq or PostgreSQL, which is realistic for a real-world application. We only consider these services hosting impact and not their development, as done by Kern *et al.* [38]. As the requirements do not specify a storage size, we used the default repository size limit of 10 GB by project, and considered 2 projects per user with a $3\times$ data redundancy; thus obtaining a total of 3,000 TB of data to store. We do not include end-users in our case study, considering that their usage environmental impact—*i.e.*, mainly through energy consumption—would be marginal in contrast to the manufacturing impact of their device, and that the property of the device does not rest on using Gitlab or not.

In this case study, we assume the development and hosting in France, and we use the corresponding impact factors.

A. Shifts between phases

As stated in Section III, the hypotheses taken when modeling and reference values used with possibly high margins of error imply uncertainty in an environmental impact assessment, to which there is no ground truth to compare. This implies caution when treating with absolute values, but stakeholders can identify impact hotspots and shifts using a high-level perspective, by manipulating orders of magnitude obtained through a common methodology and reference values.

We first focus on a specific impact category, *Climate Change* indicated in CO_2e , to show how modeling a software life cycle environmental impact can provide actionable insights

to lower its environmental impact. To respect the Paris Agreement climate objectives to limit global warming to $+1.5^\circ C$ above pre-industrial levels [52], the IPCC [32] gives a global carbon budget of 500 billion tons from 2020 to 2050—*i.e.*, roughly under two tons per year by earth inhabitants.

1) *Phases*: When applying the methodology described in Section IV with a *run* phase duration of 15 years, we obtained the distribution of tCO_2e emitted shown in Figure 3. The impact is largely dominated by the building phase, emitting approximately 13,945 tCO_2e , while the running phase is relatively low with roughly 279 tCO_2e emitted.

This highlights the importance of using a life cycle approach, as considering only the software through its usage, which is commonly done, can hide the vast majority of impact which occur during its development: more than 95% in this case.

The *build* phase and its sub-phases only consume *people* resources. As such, their environmental impact distribution is the same one as the COCOMO software effort I, where the *construction* activity dominates the man-days required, resulting in more than 64% of the *build* phase CO_2e emissions, especially for *implementation*, *assessment*, and *design* phases.

2) *Amortization*: Figure 3 highlights that in an ad-hoc development, the *build* phase can largely prevail over software's overall life cycle environmental impact. However, a project the size of Gitlab requires to be hosted for an unrealistically long duration to reach the tipping point where its *run* phase impact will be equal to the *build* phase. With a constant *build* phase emitting 13,945 tCO_2e and a year of *run* emitting 18.614 tCO_2e , this tipping point is thus identified as roughly

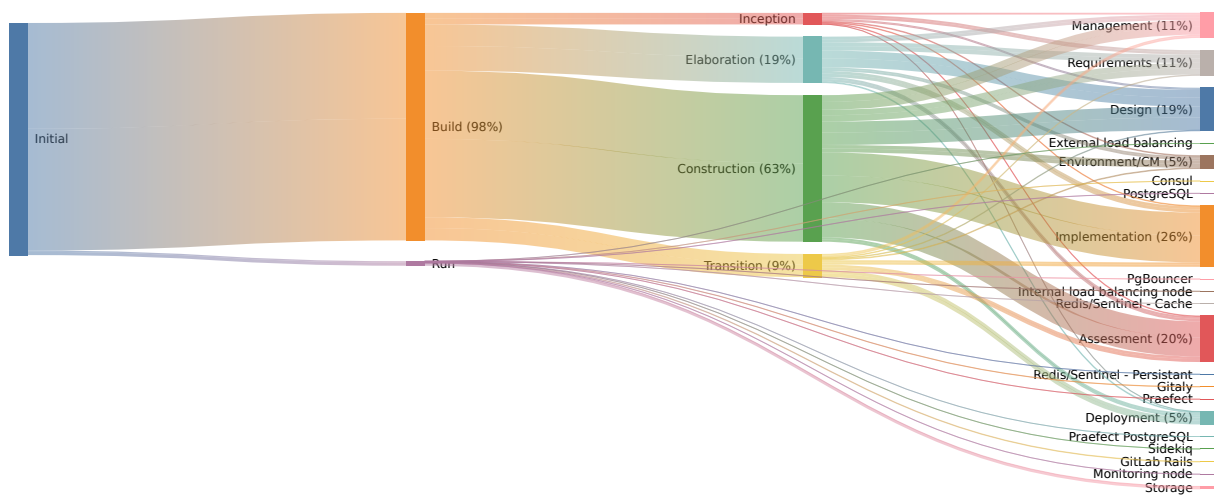


Fig. 3. CO₂e emissions distribution between life cycle phases

749 years of usage. As Gitlab is hosted simultaneously across multiple instances, we consider a *hosting-year* as a single instance hosted for a year.

B. Shifts between resources consumed

A life cycle approach allows stakeholders to identify the most impactful phases and uncover impacts hidden by focusing on a single one, but does not provide actionable insight to reduce their respective impact. As defined in Section IV a phase does not have a direct impact, it inherently comes from the resources it consumes. By using a similar holistic approach for resources, stakeholders can identify hotspots and shifts in their environmental impact.

1) *Hotspots*: The Sankey diagram of Figure 4 depicts the flows of tCO₂e emitted by the consumption of the different resources across the project life cycle. It models a project where impacts are at the tipping point between the *build* and *run* phases, previously defined and established at 749 *hosting-years*, to avoid having the *people* resource impact largely dominating the overall one.

For one day, the *people* resource emits roughly $1.21e-2$ tCO₂e and is constituted of transportation, a laptop and an external monitor, as well as an office. While the monitor and laptop impacts are relatively low, the office accounts for almost half of the impact, emitting 6,178 tCO₂e. Transportation emits the other half, 7,326 tCO₂e, based on a representative usage mix of car, motorbike, public transport and bike usage in France [55, 43] and their associated impact factors [2]. Among these, the car accounts for more than 95% of the overall commuting environmental impact and ultimately becomes one of the main impact hotspots among all resources consumed throughout the software life cycle. We thus assess Kern *et al.* [38] that commuting can be a key impact on the overall environmental impact of a software life cycle, not only on the building phase.

2) *Manufacture and usage*: Figure 4 shows that, despite the 758 vCPU and 1636 GB of memory required to host Gitlab

instances, the 300 TB of storage largely prevails over the overall CO₂e emissions. Among the 8,853 tCO₂e emitted, more than 95% comes from its manufacturing. By acknowledging only its usage impact—*i.e.*, energy consumption—it would be considered a relatively low source of impact. However, when acknowledging the hardware’s complete life cycle, its manufacturing is one of the main emissions flow, and storage becomes one of the main hotspots on the overall project life cycle. Identifying such a hotspot allows stakeholders to uncover hidden impacts, as well as prioritize actions, such as prolonging hardware life expectancy or decreasing the number of allocated resources.

3) *Compare scenarios*: To compare different approaches, Figure 2 depicts a scenario between a model at the tipping point between its *build* and *run* phases impact and a more hardware intensive one, where the overall number of *person-month* required is reduced by 20%, leading to double the hardware resources required to host an instance.

Figure 5 points out the main drivers of emissions shifts between both approaches. The decrease of *person-month* required results in cars and offices emissions lowering by almost 2,650 tCO₂e. It is however not sufficient to compensate for the increase of hardware-related ones, resulting in an overall emissions increase of 2299 tCO₂e. By modeling and comparing different scenarios, stakeholders can quickly get insights into the relative impact shifts between different implementation choices.

4) *Multicriteria*: We focused on the climate change impact category to highlight how the methodology can help stakeholders to identify impact shifts between phases and hotspots between resources, and highlight the importance of considering their life cycle impacts. However, as stated in Section II, an environmental impact is an aggregate of impact categories, and an impact can also shift between these categories.

To identify this shift, Figure 6 shows a holistic view of resources’ environmental impact along all categories of impact proposed. Due to the scarcity of open-source data, discussed in

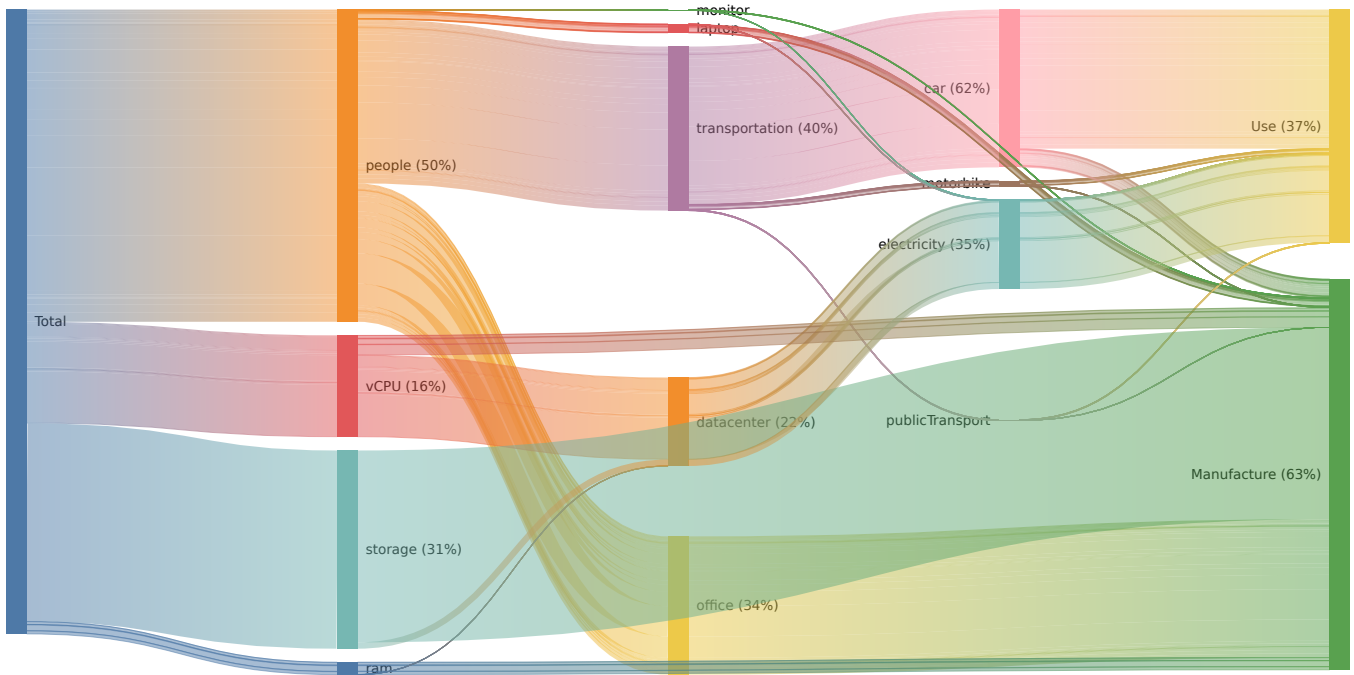


Fig. 4. CO₂e emissions distribution between resources

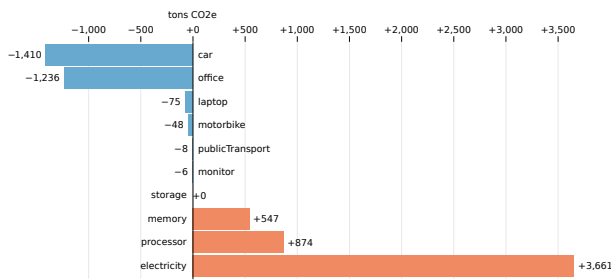


Fig. 5. Consequences of reducing the effort required by 20%

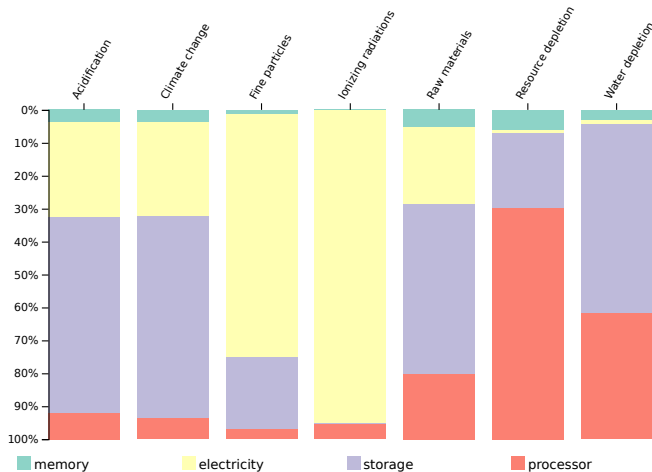


Fig. 6. Run phase resources environmental impact

Section VII, we were not able to find relevant multi-categories impact factors for transport and buildings. Thus, the figure focuses on the *use* phase, which uses hardware resources with environmental impact data sourced from [6].

Following the category of impact looked at, the order of magnitude between the main sources can vary greatly. When focusing on *climate change*, a decision such as replacing processors with newer and more energy-efficient ones can reduce their contribution to this impact category. It will however increase their contribution to other categories, such as *water depletion* and *resource depletion* with a possibly worse environmental impact. Furthermore, a holistic view of multiple categories highlights the major role of hardware manufacturing on the software hosting environmental footprint.

Figure 6 displays each category separately and in its respective unit, which can complicate the decision-making process. To better understand their relative magnitude and importance, the ISO 14044 [35] defines optional steps of an LCA: *normalization* and *weighting*. Each indicator result is transformed by dividing it by a reference value, such as the one given by Sala *et al.* [58]. This allows for identifying and prioritizing the relevant categories, as well as obtaining a single score by conducting another optional step, *grouping*. However, as there is no broadly accepted method for weighting and grouping yet [56], we chose not to aggregate impact categories and report them separated to stakeholders.

VII. THREATS TO VALIDITY

Gitlab is used as a sampled case study to show the methodology and the actionable insights it offers on a realistic dataset of a complex application, but the impact cannot be considered

as these of the project itself. Defining a scope, even with its limitation, is essential to a life cycle approach, as considering all upstream resources would be too large and thus impossible. As we limit our *build* scope to values obtained through the COCOMO effort estimation methodology, some key elements are missing for an organization, such as supporting employee infrastructure, accountancy, human resources, etc. Furthermore, the Gitlab company is *all-remote*, thus not using offices each day, which plays a key role associated with commuting in our impacts modeling. Their employees work across 65 countries, which we do not consider as we did not add work-related travel. We used a “*black-box*” approach using the source code repository and documented hardware requirements as a best-effort solution, but encourage stakeholders to use the more precise project values available to them, and impact factors better suited to their context to obtain the best insights from the model.

Our scope does not include maintenance for the software, as well as for the hardware resources which include data centers through a *Power Usage Effectiveness* (PUE) only, not with a life cycle approach that includes water usage for example. Clusters monitoring infrastructure, overhead ... are not taken into account, as well as the hardware used for development, such as test, qualification, and validation servers.

We chose not to consider *Continuous Integration / Continuous Delivery* (CI/CD) runners in our use case, as they would be predominant over the hosting impact. While relevant for the Gitlab product LCA, it is a peculiar use case, highly resource-intensive thus with a high impact. Using the default SaaS runner configuration with 1 vCPU and 3.75 GB of RAM, as well as the 10,000 minutes per month or shared runner time allocated to premium users, we obtain just about 12.77 tCO₂e per month for the runners, while the overall hosting impact is roughly 1.55 tCO₂e.

The ecosystem lacks data on the environmental impact of resources it uses, as highlighted in [49], all the more multi-criteria impact, which we demonstrate the importance of. Our model outputs are strongly correlated to its reference values, and we modeled an environmental impact using impact factors from France where the electricity is low carbon. In this context, we observed a smaller resource usage than manufacturing footprint, but this cannot be generalized and we encourage stakeholders to use impact factors corresponding to their context.

As it uses an attributional approach, our model will be limited to delivering insights into its intrinsic characteristics, but not into the side effects that might occur. An ICT product life cycle analysis will only reveal the first-order effects, not its role as an enabling technology [29]. For instance, reducing the commuting impact by encouraging employees to work from home will reduce the project’s overall impact from the model’s perspective, but teleworking may also increase the overall distance traveled as stated by Caldarola and Sorrell [10]. Offices-related impacts such as energy consumption for heating, cooling and lighting will be transferred to employees’ homes. Hook *et al.* [30] concluded through a systematic review

that economy-wide energy savings due to teleworking are typically modest, and can even be negative or nonexistent.

Our solution is not intended to conduct a full-fledged LCA to obtain the environmental impact for a software product. It however helps software stakeholders to prepare the *Life Cycle inventory* (LCI) [35] step, by providing a flow model of the software life cycle.

VIII. CONCLUSION AND NEXT STEPS

In this paper, we introduce an approach to model a software life cycle environmental impact. We show how using a bottom-up holistic approach can allow stakeholders to identify impact shifts between life cycle phases, resources consumed and categories of impact. As recommendations to lower software project impacts cannot be generalized due to their uniqueness, we propose a simple and flexible way to model project resources and activities to adapt to different development models. We publish an implementation to foster a widespread approach, and show on a sampled use case the actionable insights delivered to stakeholders by identifying impact hotspots in phases and as resources. We highlight the importance of considering software’s environmental impact over its complete life cycle, and not only its usage through energy consumption. We emphasize on the role of commuting on the overall impact, as well as the importance of considering ICT hardware resources’ complete life cycle, as their manufacturing impact can be larger than their usage one. We encourage stakeholders to consider multiple impact categories, to avoid shiftings by focusing on a single one.

Based on our work and due to the scarcity of reference data to model software environmental impact, we suggest that future work explore the ICT hardware impact along multiple impact categories, especially its propagation through abstraction layers.

ACKNOWLEDGMENTS

This work is partly supported by the ANR Distiller grant (ANR-21-CE25-0022).

REFERENCES

- [1] SDIA - Digital Carbon Footprint Steering Group (SG). *Methodology for calculating the environmental footprint of server-side applications*. [Online; accessed 24. Jan. 2023]. 2022. URL: <https://github.com/SDIAlliance/carbon-footprint-ssa>.
- [2] ADEME - *Site Bilans GES*. [Online; accessed 14. Feb. 2023]. Feb. 2023. URL: <https://bilans-ges.ademe.fr/fr/basecarbone/donnees-consulter/>.
- [3] AIDanial. *cloc*. [Online; accessed 17. Jan. 2023]. Jan. 2023. URL: <https://github.com/AIDanial/cloc>.
- [4] Anders S. G. Andrae and Tomas Edler. “On Global Electricity Usage of Communication Technology: Trends to 2030”. In: *Challenges* 6.1 (2015), pp. 117–157. ISSN: 2078-1547. DOI: 10.3390/challe6010117. URL: <https://www.mdpi.com/2078-1547/6/1/117>.

- [5] Anders SG Andrae. “New perspectives on internet electricity use in 2030”. In: *Engineering and Applied Science Letters* 3.2 (2020), pp. 19–31. DOI: 10.30538/psrp-easl2020.0038.
- [6] *Base Impacts - Documentation - Numérique(Négaocet)*. [Online; accessed 14. Feb. 2023]. Feb. 2023. URL: <https://base-impacts.ademe.fr/documents/Numerique.zip>.
- [7] Lotfi Belkhir and Ahmed Elmeligi. “Assessing ICT global emissions footprint: Trends to 2040 & recommendations”. In: *Journal of Cleaner Production* 177 (2018), pp. 448–463. ISSN: 0959-6526. DOI: <https://doi.org/10.1016/j.jclepro.2017.12.239>. URL: <https://www.sciencedirect.com/science/article/pii/S095965261733233X>.
- [8] Pankaj Bhatia et al. “Greenhouse gas protocol product life cycle accounting and reporting standard”. In: (2011). URL: https://ghgprotocol.org/sites/default/files/standards/Product-Life-Cycle-Accounting-Reporting-Standard_041613.pdf.
- [9] Barry Boehm et al. “Cost models for future software life cycle processes: COCOMO 2.0”. In: *Annals of Software Engineering* 1.1 (Dec. 1995), pp. 57–94. ISSN: 1573-7489. DOI: 10.1007/BF02249046. URL: <https://doi.org/10.1007/BF02249046>.
- [10] Bernardo Caldarola and Steve Sorrell. “Do teleworkers travel less? Evidence from the English National Travel Survey”. In: *Transportation Research Part A: Policy and Practice* 159 (2022), pp. 282–303. ISSN: 0965-8564. DOI: <https://doi.org/10.1016/j.tra.2022.03.026>. URL: <https://www.sciencedirect.com/science/article/pii/S0965856422000738>.
- [11] A. Carette et al. “Investigating the energy impact of Android smells”. In: *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. Los Alamitos, CA, USA: IEEE Computer Society, Feb. 2017, pp. 115–126. DOI: 10.1109/SANER.2017.7884614. URL: <https://doi.ieeecomputersociety.org/10.1109/SANER.2017.7884614>.
- [12] Chien A. Chan et al. “Methodologies for Assessing the Use-Phase Power Consumption and Greenhouse Gas Emissions of Telecommunications Network Services”. In: *Environmental Science & Technology* 47.1 (2013). PMID: 23211093, pp. 485–492. DOI: 10.1021/es303384y. eprint: <https://doi.org/10.1021/es303384y>. URL: <https://doi.org/10.1021/es303384y>.
- [13] Arti Chandani, Smita Waghlikar, and Om Prakash. “A Bibliometric Analysis of Green Computing”. In: *Information and Communication Technology for Competitive Strategies (ICTCS 2021)*. Ed. by Amit Joshi, Mufti Mahmud, and Roshan G. Ragel. Singapore: Springer Nature Singapore, 2023, pp. 547–557. ISBN: 978-981-19-0095-2.
- [14] Intergovernmental Panel on Climate Change. “Carbon dioxide-equivalent (CO₂-eq) emissions and concentrations” in *Topic 2 of the Synthesis Report*. 2007. URL: <https://www.ipcc.ch/report/ar4/syr/>.
- [15] Vlad C. Coroama and Lorenz M. Hilty. “Assessing Internet energy intensity: A review of methods and results”. In: *Environmental Impact Assessment Review* 45 (2014), pp. 63–68. ISSN: 0195-9255. DOI: <https://doi.org/10.1016/j.ear.2013.12.004>. URL: <https://www.sciencedirect.com/science/article/pii/S0195925513001121>.
- [16] Markus Dick and Stefan Naumann. “Enhancing Software Engineering Processes towards Sustainable Software Product Design”. In: *Integration of Environmental Information in Europe: Proceedings of the 24th International Conference on Informatics for Environmental Protection, EnviroInfo 2010, Cologne/Bonn, Germany, October 6-8, 2010*. Ed. by Klaus Greve and Armin B. Cremers. Shaker Verlag, Aachen, 2010, pp. 706–715. URL: <http://iai-uiserv1.iai.fzk.de/ictensure/site?mod=litdb%5C&subject=art%5C&pid=X167A957D%5C&action=detail>.
- [17] United Nations Statistics Division. — *SDG Indicators*. [Online; accessed 27. Sep. 2022]. Sept. 2022. URL: <https://unstats.un.org/sdgs/report/2019/goal-12>.
- [18] T. Ekvall. “Cleaner production tools: LCA and beyond”. In: *Journal of Cleaner Production* 10.5 (2002). Integrating greener product development perspectives, pp. 403–406. ISSN: 0959-6526. DOI: [https://doi.org/10.1016/S0959-6526\(02\)00026-4](https://doi.org/10.1016/S0959-6526(02)00026-4). URL: <https://www.sciencedirect.com/science/article/pii/S0959652602000264>.
- [19] Matthias Finkbeiner et al. “The new international standards for life cycle assessment: ISO 14040 and ISO 14044”. In: *The international journal of life cycle assessment* 11.2 (2006), pp. 80–85.
- [20] Alcides Fonseca, Rick Kazman, and Patricia Lago. “A Manifesto for Energy-Aware Software”. In: *IEEE Software* 36.6 (2019), pp. 79–82. DOI: 10.1109/MS.2019.2924498.
- [21] Charlotte Freitag et al. “The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations”. In: *Patterns* 2.9 (2021), p. 100340. ISSN: 2666-3899. DOI: <https://doi.org/10.1016/j.patter.2021.100340>. URL: <https://www.sciencedirect.com/science/article/pii/S2666389921001884>.
- [22] *General Principles for the Environmental Labelling of Consumer Products, Methodological Standard for the Environmental Assessment of Digital Service*. Tech. rep. [Online; accessed 2. Feb. 2023]. ADEME, July 2021. URL: <http://www.base-impacts.ademe.fr/documents/Numerique.zip>.
- [23] *GitLab.org / GitLab · GitLab*. [Online; accessed 17. Jan. 2023]. Jan. 2023. URL: <https://gitlab.com/gitlab-org/gitlab>.
- [24] *Green data centers in focus*. [Online; accessed 5. Jul. 2022]. Dec. 2021. URL: http://english.www.gov.cn/statecouncil/ministries/202112/09/content_WS61b13edac6d09c94e48a1f81.html.

- [25] Green-Software-Foundation. *software_carbon_intensity*. [Online; accessed 24. Jan. 2023]. Jan. 2023. URL: https://github.com/Green-Software-Foundation/software_carbon_intensity/blob/dev/Software_Carbon_Intensity/Software_Carbon_Intensity_Specification.md.
- [26] Udit Gupta et al. “Chasing Carbon: The Elusive Environmental Footprint of Computing”. In: *CoRR* abs/2011.02839 (2020). arXiv: 2011.02839. URL: <https://arxiv.org/abs/2011.02839>.
- [27] *Handbook on Life Cycle Assessment*. Springer Netherlands. ISBN: 978-0-306-48055-3. URL: <https://link.springer.com/book/10.1007/0-306-48055-7>.
- [28] hgrecco. *pint*. [Online; accessed 14. Feb. 2023]. Feb. 2023. URL: <https://github.com/hgrecco/pint>.
- [29] Lorenz M Hilty. “CO2 Reduction with ICT: Prospects and Barriers.” In: *EnviroInfo* (1). 2007, pp. 35–42.
- [30] Andrew Hook et al. “A systematic review of the energy and climate impacts of teleworking”. In: *Environmental Research Letters* 15.9 (Aug. 2020), p. 093003. DOI: 10.1088/1748-9326/ab8a84. URL: <https://dx.doi.org/10.1088/1748-9326/ab8a84>.
- [31] IPCC. *Climate Change 2022: Mitigation of Climate Change. Contribution of Working Group III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Ed. by P.R. Shukla et al. Cambridge, UK and New York, NY, USA: Cambridge University Press, 2022. DOI: 10.1017/9781009157926.
- [32] IPCC. “Mitigation Pathways Compatible with 1.5°C in the Context of Sustainable Development”. In: *Global Warming of 1.5°C: IPCC Special Report on Impacts of Global Warming of 1.5°C above Pre-industrial Levels in Context of Strengthening Response to Climate Change, Sustainable Development, and Efforts to Eradicate Poverty*. Cambridge University Press, 2022, pp. 93–174. DOI: 10.1017/9781009157940.004.
- [33] IEC ISO et al. “ISO/IEC/IEEE 12207: 2017, Systems and software engineering—Software life cycle processes”. In: *International Organization for Standardization* (2017).
- [34] *ISO 14040:2006*. [Online; accessed 18. Jul. 2022]. July 2022. URL: <https://www.iso.org/fr/standard/37456.html>.
- [35] *ISO 14044:2006*. [Online; accessed 7. Feb. 2023]. Feb. 2023. URL: <https://www.iso.org/fr/standard/38498.html>.
- [36] ITU-T. *L.1410 : Methodology for environmental life cycle assessments of information and communication technology goods, networks and services*. [Online; accessed 2. Feb. 2023]. Dec. 2014. URL: <https://www.itu.int/rec/T-REC-L.1410-201412-I/en>.
- [37] Erik A. Jagroep et al. “Software Energy Profiling: Comparing Releases of a Software Product”. In: *Proceedings of the 38th International Conference on Software Engineering Companion*. ICSE ’16. Austin, Texas: Association for Computing Machinery, 2016, pp. 523–532. ISBN: 9781450342056. DOI: 10.1145/2889160.2889216. URL: <https://doi.org/10.1145/2889160.2889216>.
- [38] Eva Kern et al. “Impacts of software and its engineering on the carbon footprint of ICT”. In: *Environmental Impact Assessment Review* 52 (2015), pp. 53–61.
- [39] Aziz Deraman Komeil Raisian Jamaiah Yahaya. “Green Measurements for Software Product Based on Sustainability Dimensions”. In: *Computer Systems Science and Engineering* 41.1 (2022), pp. 271–288. DOI: 10.32604/csse.2022.020496. URL: <http://www.techscience.com/csse/v41n1/44795>.
- [40] Mohankumar Kumar. “A GREEN IT STAR MODEL APPROACH FOR SOFTWARE DEVELOPMENT LIFE CYCLE”. In: *international journal of Advanced technology in engineering and Science* 3 (Mar. 2015), pp. 548–559.
- [41] P. Lago, Q. Gu, and P. Bozzelli. *A systematic literature review of green software metrics*. VU Technical Report, 2014.
- [42] Giuseppe Lami, Fabrizio Fabbri, and Mario Fusani. “Software Sustainability from a Process-Centric Perspective”. In: *Systems, Software and Services Process Improvement*. Ed. by Dietmar Winkler, Rory V. O’Connor, and Richard Messnarz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 97–108. ISBN: 978-3-642-31199-4.
- [43] *Les cadres parcourent chaque année près de 3000 km de plus que les employés pour se rendre au travail | L’Observatoire des Territoires*. [Online; accessed 14. Feb. 2023]. Feb. 2023. URL: <https://www.observatoire-des-territoires.gouv.fr/kiosque/2019-mobilite-10-les-cadres-parcourent-chaque-annee-pres-de-3000-km-de-plus-que-les>.
- [44] Sara S Mahmoud and Imtiaz Ahmad. “A green model for sustainable software engineering”. In: *International Journal of Software Engineering and Its Applications* 7.4 (2013), pp. 55–74.
- [45] Jens Malmodin and Dag Lundén. “The Energy and Carbon Footprint of the Global ICT and E&M Sectors 2010-2015”. In: *Sustainability* 10.9 (2018). ISSN: 2071-1050. DOI: 10.3390/su10093027. URL: <https://www.mdpi.com/2071-1050/10/9/3027>.
- [46] Eric Masanet et al. “Recalibrating global data center energy-use estimates”. In: *Science* 367.6481 (2020), pp. 984–986. DOI: 10.1126/science.aba3758. eprint: <https://www.science.org/doi/pdf/10.1126/science.aba3758>. URL: <https://www.science.org/doi/abs/10.1126/science.aba3758>.
- [47] Florin-Constantin Mihai et al. “Chapter 1 - Waste Electrical and Electronic Equipment (WEEE): Flows, Quantities, and Management—A Global Scenario”. In: *Electronic Waste Management and Treatment Technology*. Ed. by Majeti Narasimha Vara Prasad and Meththika Vithanage. Butterworth-Heinemann, 2019, pp. 1–34. ISBN: 978-0-12-816190-6. DOI: <https://doi.org/10.1016/B978-0-12-816190-6.00001-7>. URL: <https://doi.org/10.1016/B978-0-12-816190-6.00001-7>.

<https://www.sciencedirect.com/science/article/pii/B9780128161906000017>.

- [48] Sambit Kumar Mishra et al. “Energy-efficient VM-placement in cloud data center”. In: *Sustainable Computing: Informatics and Systems* 20 (2018), pp. 48–55. ISSN: 2210-5379. DOI: <https://doi.org/10.1016/j.suscom.2018.01.002>. URL: <https://www.sciencedirect.com/science/article/pii/S2210537917302536>.
- [49] David Mytton. “Assessing the suitability of the Greenhouse Gas Protocol for calculation of emissions from public cloud computing workloads”. en. In: *Journal of Cloud Computing* 9.1 (Dec. 2020), p. 45. ISSN: 2192-113X. DOI: 10.1186/s13677-020-00185-8. URL: <https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-020-00185-8> (visited on 12/17/2021).
- [50] Stefan Naumann et al. “The GREENSOFT Model: A reference model for green and sustainable software and its engineering”. en. In: *Sustainable Computing: Informatics and Systems* 1.4 (Dec. 2011), pp. 294–304. ISSN: 22105379. DOI: 10.1016/j.suscom.2011.06.004. URL: <https://linkinghub.elsevier.com/retrieve/pii/S2210537911000473> (visited on 11/25/2021).
- [51] Zakaria Ournani et al. “On Reducing the Energy Consumption of Software: From Hurdles to Requirements”. In: *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. ESEM '20. Bari, Italy: Association for Computing Machinery, 2020. ISBN: 9781450375801. DOI: 10.1145/3382494.3410678. URL: <https://doi.org/10.1145/3382494.3410678>.
- [52] *Paris Agreement*. [Online; accessed 11. Feb. 2023]. Dec. 12, 2015. URL: https://treaties.un.org/pages/ViewDetails.aspx?src=TREATY&mtdsg_no=XXVII-7-d&chapter=27&clang=en.
- [53] GHG Protocol. “ICT Sector Guidance built on the GHG Protocol Product Life Cycle Accounting and Reporting Standard”. In: *Global: GHG Protocol* (2017).
- [54] *Reference architecture: up to 50,000 users | GitLab*. [Online; accessed 23. Jan. 2023]. Jan. 2023. URL: https://docs.gitlab.com/ee/administration/reference_architectures/50k_users.html.
- [55] *Résultats détaillés de l'enquête mobilité des personnes 2019*. [Online; accessed 14. Feb. 2023]. Feb. 2023. URL: <https://www.statistiques.developpement-durable.gouv.fr/resultats-detailles-de-lenquete-mobilite-des-personnes-de-2019?rubrique=60&dossier=1345>.
- [56] Andreas Roesch, Serenella Sala, and Niels Jungbluth. “Normalization and weighting: the open challenge in LCA”. In: *The International Journal of Life Cycle Assessment* 25.9 (Sept. 2020), pp. 1859–1865. ISSN: 1614-7502. DOI: 10.1007/s11367-020-01790-0. URL: <https://doi.org/10.1007/s11367-020-01790-0>.
- [57] Nayan B. Ruparelia. “Software Development Lifecycle Models”. In: *SIGSOFT Softw. Eng. Notes* 35.3 (May 2010), pp. 8–13. ISSN: 0163-5948. DOI: 10.1145/1764810.1764814. URL: <https://doi.org/10.1145/1764810.1764814>.
- [58] Sala S et al. “Global normalisation factors for the Environmental Footprint and Life Cycle Assessment”. In: KJ-NA-28984-EN-C (print),KJ-NA-28984-EN-N (online) (2017). ISSN: 1018-5593 (print),1831-9424 (online). DOI: 10.2760/88930(online),10.2760/775013(print).
- [59] Björn A. Sandén and Magnus Karlström. “Positive and negative feedback in consequential life-cycle assessment”. In: *Journal of Cleaner Production* 15.15 (2007), pp. 1469–1481. ISSN: 0959-6526. DOI: <https://doi.org/10.1016/j.jclepro.2006.03.005>. URL: <https://www.sciencedirect.com/science/article/pii/S0959652606001442>.
- [60] Daniel Schien et al. “The Energy Intensity of the Internet: Edge and Core Networks”. In: *ICT Innovations for Sustainability*. Ed. by Lorenz M. Hilty and Bernard Aebischer. Cham: Springer International Publishing, 2015, pp. 157–170. ISBN: 978-3-319-09228-7.
- [61] Sanath. S. Shenoy and Raghavendra Eeratta. “Green software development model: An approach towards sustainable software development”. In: *2011 Annual IEEE India Conference*. 2011, pp. 1–6. DOI: 10.1109/INDCON.2011.6139638.
- [62] Sanath. S. Shenoy and Raghavendra Eeratta. “Green software development model: An approach towards sustainable software development”. In: *2011 Annual IEEE India Conference*. 2011, pp. 1–6. DOI: 10.1109/INDCON.2011.6139638.
- [63] Md Abu Bakar Siddik, Arman Shehabi, and Landon Marston. “The environmental footprint of data centers in the United States”. en. In: *Environmental Research Letters* 16.6 (June 2021), p. 064017. ISSN: 1748-9326. DOI: 10.1088/1748-9326/abfba1. URL: <https://iopscience.iop.org/article/10.1088/1748-9326/abfba1> (visited on 07/18/2022).
- [64] Juha Taina. “How green is your software?” In: *International Conference of Software Business*. Springer. 2010, pp. 151–162.
- [65] Roberto Verdecchia, Patricia Lago, and Carol de Vries. “The future of sustainable digital infrastructures: A landscape of solutions, adoption factors, impediments, open problems, and scenarios”. In: *Sustainable Computing: Informatics and Systems* 35 (2022), p. 100767. ISSN: 2210-5379. DOI: <https://doi.org/10.1016/j.suscom.2022.100767>. URL: <https://www.sciencedirect.com/science/article/pii/S2210537922000889>.
- [66] *Waste electrical and electronic equipment (WEEE) by waste management operations - open scope, 6 product categories (from 2018 onwards) - Products Datasets - Eurostat*. [Online; accessed 27. Sep. 2022]. Sept. 2022. URL: https://ec.europa.eu/eurostat/web/products-datasets/-/env_waseleeeos.
- [67] Henrik Wenzel. “Application dependency of lca methodology: Key variables and their mode of influenc-

ing the method". In: *The International Journal of Life Cycle Assessment* 3.5 (Sept. 1998), pp. 281–288. ISSN: 1614-7502. DOI: 10.1007/BF02979837. URL: <https://doi.org/10.1007/BF02979837>.