# A Sustainability Matrix for Smartphone Application

Amal Alsayed, Rébecca Deneckère

# A sustainability matrix for smartphone application

Amal Alsayed[1] and Rébecca Deneckère[2]

[1] RTE (Réseau de Transport d'Électricité), Paris la défense
[2] Centre de Recherche en Informatique, University Paris 1 Panthéon-Sorbonne, Paris, France
amal.alsayed90@gmail.com, Rebecca.Deneckere@univ-paris1.fr

**Abstract.** Nowadays, smartphones are an indispensable part of most people's lives, they are a means of enabling communication and exchange that makes everyone comes together. However, it comes with a cost as the production and use of these systems is affecting the global earth ecosystem. Considerations about sustainability are more and more in people minds and the possibility to offer sustainable smartphone applications can be an asset to any company. We propose in this work a sustainability matrix for Smartphone in order to identify their sustainability degree and offer some new thinking to their developers. We created this matrix with a state of the art study, expert interviews and a public questionnaire. We evaluate our matrix on an application and discuss the results with its development project manager.

**Keywords:** Sustainability, Smartphone, Application, Criteria, Matrix.

## 1 Introduction

Technologies are progressing considerably to meet the ever-increasing needs of humans. However, we now that this quest for progress has harmful effects on the environment and therefore on humanity. One of the most threatening impacts is ecological.

A large majority of scientists agree on the severity of climate change [1]. Increase in the average temperature at the globe surface, decrease in soil fertility, ice melting at the poles are effects which must be limited as far as possible and as soon as possible. There is clear evidence that information technologies contribute to this environmental impact and climate change [2]. Living a smart life comes now with a smart way to handle the life cycle of technological products in the best possible way.

The strategy followed by companies to reduce these effects is called Green IT. However, even if green IT presents huge potential on the corporate level [3], there is little research regarding its application and the potential outside the organizations [4]. In this field, it has been shown that the production of products traditionally represents a factor of 100 in relation to their use. We will address the issue of sustainable use of hardware through sound software management. If the environmental issue is based on making a material sustainable, then we can wonder if the software and its life cycle cause the premature renewal of terminals. [5] states that *"Most of the figures available today show that digital power is responsible for 3.7% of the world's total greenhouse gas (GHG) emissions in 2018 and 4.2% of global primary energy consumption. Worldwide, 44% of this footprint is due to the manufacture of terminals, computer*

*centers and networks and 56% to their use."* The equipment life cycle consists of several steps that begin with prefabrication and end with the end of life of the equipment [6]. Each step has its own impact on the environment. The prefabrication phase has the greatest impact on the environment, since it requires the extraction of non-renewable materials, followed by the use phase [6]. The latter has an impact on the environment because of the induced energy consumption.

We believe that software can cause the premature renewal of terminals. This comes in many forms, like the disproportionate demand for machine resources (due to poor design and/or software development), the impact of software and the demands of associated hardware devices (misuse) or a contribution to programmed obsolescence (i.e. mercantile in this case).

Considering the average population of a developed country and its collective unconsciousness, people believe that software, due to its lack of physical existence, would have no direct impact on the environment. To demonstrate the opposite, it is enough to recall that the hardware stores information (hard drive, USB drive), and runs the software that operates calculations to render services (processors, memory, etc.). These elements have an impact on the environment at every stage of their life cycle. Therefore, to accept this hypothesis or reject it, we must measure the influence of software to determine its impact on equipment and therefore environmental impacts.

Software sustainability can be read at first as an oxymoron. Indeed, the turnover of technologies, hardware, etc. is such today that it may seem vain to want to make a software sustainable. On the one hand, the definition of sustainability focuses more on the environmental impact than on the objective of making software last for a long time. On the other hand, some works shows that it is possible to design a software incorrectly, to develop it incorrectly, to use it incorrectly and to leave a strong residual imprint [7][8].

The purpose of this paper is therefore to propose a software sustainability matrix that outlines the criteria for software sustainability. Ideally, this matrix will allow us to determine whether a software is sustainable or not. There are several families of software but we restricted our work mainly on the SmartPhones software family.

Section 2 gives some information about the background. Our research methodology is explained in section 3 with an identification of the sources of our defined criteria. Our proposed sustainability matrix is then completely defined in Section 4 and evaluated in Section 5. We conclude in Section 6.

## 2 Background

One of the most common interrogation nowadays is the problem of how to preserve the environment. How can we reduce greenhouse gas pollution and increase the percentage of carbon dioxide in the air? In this context, the concept of Sustainability has emerged among many efforts. It has been applied in several areas such as real estate [9], home appliances [10], marketing [11], transportation [12] and so on. Of course, one of those areas is information technology, which in turn tries to reduce its negative impact on the environment [13].

Sustainable Management is the long-term process of optimizing environmental, economic and social performance simultaneously while taking into account natural resource restrictions, This allows business to continue without compromising the needs of future generations [14].

It is necessary to make a clear distinction between Information Technology and Information Systems (IS). *Information Technology* refers to all that is computer hardware, software or peripheral information equipment. *Green Information Technology* refers to measures and initiatives that aim to reduce the negative environmental impact resulting from the manufacture, operation and disposal of IT equipment and infrastructure [14]. *Information System*, on the other hand, is a broader concept that covers both technological components and human activities related to the process of managing and using technology within the organization. Information technology affects our environment in many different ways. Every stage of a computer's life, from its production to its disposal and use, poses environmental problems. Each CP used generates approximately one ton of carbon dioxide each year [15]. *Green Information Systems* refers to practices that aim to reduce the negative environmental impacts of IS, business operations and IS-based products and services. All this by determining how we use to invest, deploy, use and manage IS information systems [14].

We have to better understand the environmental impacts of information technology and how to make our IT infrastructures, products, services, operations, applications and practices environmentally friendly. [15] proposes to apply Green IT on four several ways : Green use (to reduce the energy consumption of computers and other IS), Green disposal (to refurbish, reuse or recycle old computers), Green design (to design energy efficient and environmentally sound components, computers, servers, and cooling equipment), and Green manufacturing (to manufacture electronic components with minimal or no impact on the environment).

Talking about software durability makes no sense if one does not consider the durability of the hardware on which they are instantiated. The ecological footprint depends on the durability of the equipment. There are many types of computer hardware, such as computers, smartphones, printers, video game consoles, televisions connected to a box, connected objects, etc. Each type has an ecological footprint.

There are several studies that clarify the ecological footprint concept. One of these studies is the GreenIT.fr team's Global Digital Environmental Footprint [16], which focuses on quantifying the global digital environmental footprint and its evolution between 2010 and 2025. This study applies to all electronic equipment that manipulates binary data. The methodology followed is based on Life Cycle Analysis (LCA), and has quantified environmental impacts based on three categories (users, networks, computer centres) that are aggregated by a meta-model. The four indicators considered in this study are: [16] Abiotic Resource Depletion (ADP) (impact of technology on the depletion of mineral stocks), Global Warming (GHG) (climate change), Energy balance (EP) (The so-called "primary" energy is the energy required to produce the final energy. So in digital as in any other industry, in order to manufacture the equipment and depending on the stage of its life cycle, there will be different types of primary energy to produce the final energy needed to achieve that stage), Tension on Fresh Water (Water) (After breathing air, fresh water is classified as the second most important basic physiological resource also for humans than for other life forms. It should be noted that

only blue water is accessible to humans, unlike green water, which is only accessible to plants. Therefore, if digital technology increases the consumption of blue water, it will be less available for other uses at some point).

As Anne-Cécile Orgerie explains [2], ICTs and all computer equipment account for between 2 and 10% of carbon dioxide emissions according to the studies, with an agreement around 4-105%. However, aviation accounts for about 2% of carbon dioxide emissions. Studies show that both manufacturing and recycling phases are responsible for the highest proportion of carbon dioxide emissions. Hence the importance of coupling between hardware and software to develop durable software to make hardware more durable with the longest possible period of use.

Software durability alone is not enough to make hardware durable. At the same time, there are also prerequisites for the durability of the equipment. It appeared that the software, by its poor design, makes the computer hardware obsolete faster than it is expected [7]. From a general perspective, the life cycle of computer hardware can be segmented into five major phases: [6] Prefabrication, Manufacture, Distribution and Transport, Use, End of life. All the phases are important when talking about sustainability [17].

Some works exists on environmental sustainable mobile applications, as stated in a literature review presented in [18]. Some applications provide feedbacks about sustainability information on an application but are restricted to specific domains, whereas others facilitates sustainable behaviors. [18] categorized the identified applications by their goals and functionalities to provide an overview of existing user-centric Green IS solutions from IS research and practice. They assessed the potentials of mobile applications to contribute to environmental sustainability and provide a holistic perspective by performing an extensive classification of existing apps.

Applications like Ecometer[1], GTmetric[2] or EcoIndex[3] for instance, proposes a free analyze of a website sustainability. Quentic[4] proposes a customizable sustainability software to track and manage environmental compliance online. All these services use some criteria that can be rapidly characterized in order to provide a quick feedback.

## 3 Research methodology

Every phase of terminal life cycle, from manufacturing to construction or recycling, has an impact on the environment. This is why Green IT is responsible for studying each phase separately and finding more sustainable solutions. We will restrict our work on this research on the two first phases of software lifecycles. Once the research question was identified, in order to identify the more important criteria for smartphone software sustainability, we conducted a state of art literature review, interviews with experts and launched a questionnaire. We then used the found criteria to create a sustainability matrix that we evaluated on a specific application (figure 1).

---

[1] http://www.ecometer.org/

[2] https://gtmetrix.com/

[3] http://www.ecoindex.fr/

[4] https://www.quentic.com/sustainability-software/


**Fig. 1.** Research medthodology

## 3.1 Identify research question

Our hypothesis is that the lifecycle of a software can cause the renewal of the hardware. We stated that we have to measure the influence of the said software to determine its impact on equipment and therefore environmental impacts. Our research question is then the following: **Is it possible to evaluate a software, based on predefined criteria, to identify its sustainability?**. We will specifically address in this work the software family of smartphones and the two first phases of a software lifecycle.

## 3.2 Literature study: State of art

A lot of works proposes some criteria of sustainability for information systems, softwares or websites.

Pang *et al.* conducted a survey presented in the article "What Do Programmers Know about Software Energy Consumption? [19]', which revealed that programmers had limited knowledge of energy efficiency, were not aware of best practices for reducing software energy consumption and were not sure how software consumes energy. This issue of a non-functional requirement becomes very important with the growing popularity of mobile computing and the emergence of large-scale cloud deployments, and knowing that while energy consumption at the individual level was negligible, it would not be the case on a global scale because the energy consumed by all mobile devices and data centers is multiplying.

The energy behavior of a smartphone has been studied in detail by an empirical study with laboratory measurements by C. al. [20] According to the results of this study, the biggest energy consumers are the GSM module and the display, including the LCD panel, the touch screen, the accelerator/ graphics driver and the backlight. The results of the study also indicate that audio consumes a large portion of static consumption, in the range of 28-34mW. Overall, according to the study, RAM, audio, and SD card have little effect on the electrical consumption of the device and therefore offer little potential for energy optimization.

[7] proposes a practical guide to think about sustainable computing holistically, starting with the choices you make when buying technology, through to the software and peripherals you use, how you store and work with information, manage your security, save power, and maintain and dispose of your old hardware. It looks at the use of IT from the viewpoint of the information which the system manipulates.

[21] propose a label called EcoSoft that takes into account the involvement of stakeholders (project manager, software architects, developers) in the process of integrating sustainability into the project. The three stages of the software's life cycle, namely the development, usage and end of life, are analyzed to determine the environmental impacts they generate. The analysis focus on the energy consumption of software components, which is an important aspect for the overall quality of the software, especially for the user experience on the mobile device, but also because digital energy consumption has a high environmental footprint.

[8] proposes a review about Energy Consumption & IT systems, starting from the view-point of Green IT. They introduce a taxonomy of concepts, present recent data on energy consumption trends and some guidelines to write energy efficient software.

We identified several criteria in the literature, as shown in the table 1.

## 3.3 Qualitative study: Interviews

Seven experts working at RTE in the R&D and DSIT departments were selected to be interviewed. Questions asked were fairly open to cover as much as possible aspects related to the issue raised. The interviews, which were rich in information, made it possible to build up and strengthen the research axis. They also focused on the main points to be raised when dealing with the subject of Green-IT.

During these interviews, several definitions of the Green-IT concepts were clarified, and several substantive criteria to optimize the performance of the software were determined in order to visualize their impacts on the sustainability of the latter (cf. table 1).

One criteria that popped up in the interviews as a very high concern about sustainability is the programmed obsolescence (the fact that some applications are designed to be "out of date" when the devices are getting older), also widely discussed in [7]. However, this criteria is not so easy to identify and we decided not to include it in our matrix. Another criteria concerning the other phases of the lifecycle appears in the answers, like an excessive use of the battery or the performance on devices.

## 3.4 Quantitative Study: Questionnaire

A questionnaire was conducted to collect the criteria that make the software more sustainable or not according to the developers. Two versions were conducted, one in French and the other in English. There were two main sections. The first provides information on respondents' years of experience and programming languages. It also lets you know whether respondents are smartphone app developers or not. Because this questionnaire was addressed first to smartphone developers, and then to all developers. The second allows to know if the developers take into account the energy consumption, and to collect the criteria that allow to build the sustainability matrix. In addition, this

**Table 1.** Identified criteria

| Criteria | SoA | Interviews | Quest. |
|---|---|---|---|
| Separation between security updates and functional updates | | | 52,2% |
| Lazy loading | | X | 52,2% |
| Object Oriented vs Functional Programming | | X | |
| Cloud Synchronization | [8] [22] [23] | X | |
| Database VS file | [7] | X | |
| Local VS Network Storage | [7] | X | |
| Network Call | [7] | X | |
| Compiled VS Interpreted | [19] | X | |
| Background work | [19] | | 47,8% |
| Automatically launch on default startup | | | 39,1% |
| Night/Day Mode | | X | 43,5% |
| Optimize the use of the CPU | [19] | | 39,1% |
| Optimizing Algorithms (Human Action) | [21] | X | |
| Optimize code instructions (Compiler action) | [21] | X | |
| I/O RAM VS Hard Drive Rate | [8] [24] | X | |
| Optimizing the use of the Memory | [8] | | 21,7% |
| Binary that takes up space | | X | 4,3% |
| Percentage of use of Open Source | [25] [7] | | 13% |
| Bugs | | | 20% |
| The poor legibility of the code to better understand it (Evolution correction) | | X | |

section includes two open-ended questions for developers to suggest ways to assess the sustainability of software and technical design methods. 33 persons answered the questionnaire. The results showed that developers had different levels of experience with a higher representation of more than 10 years and 3 to 5 years of experience. In addition, the most commonly used programming languages are: Java, C++ and Python, with a lower representation of: Objective C, Swift and KotLin.

The difference in the level of experience of the developers allows to observe that the participants do not have the same knowledge of how the software consumes the resources of the machine and, consequently, the awareness of energy consumption. This is also confirmed by the results of the fourth question: Do you take into account energy consumption when developing software? If so, how do you account for energy consumption? Most of the answers to the fourth question showed a definite interest in this problem, but a difficult implementation.

Some criteria were identified with the questionnaire results. For instance the lazy loading and the attachment of security updates to functional updates are of utmost importance for our experts to make a software more sustainable. Intensive use of CPU or of memory is also of critical importance for the renewal of terminals. The criteria found in the questionnaire are indicated in table 1 with the percentage of respondents who identified these criteria in their answers.

The last two questions were open-ended questions for developers to see if they were aware of ways to assess sustainability, a technical design that promotes software sustainability. These answers allowed to see that developers are not familiar with the

right design and development techniques, nor the ways software damages hardware. In addition, some companies do not consider design and sustainability in their strategies.

# 4 Sustainability matrix

We defined a maturity matrix where the software score identify the sustainability degree of a software (From 1 for a bad sustainability to 5, for an excellent one). Each criteria is a characteristic of either the design phase or the development phase. They are associated to a weight identified by their appearance in the literature, the interviews and the questionnaire.

## 4.1 Design sustainability criteria

Design sustainability is composed of 11 criteria.

**Separation between security updates and functional updates (Weight: 30)**
It is known that partial updates require significantly less energy for all technologies [26]. It is then useful to identify which kind of updates are strictly necessary and the others. This criterion means that the software publisher must be transparent in separating security updates from functional updates by giving the customer the choice of accepting security updates and accepting or refuting functional updates. Because security updates are essential for the proper disposal of the software, unlike functional updates, which are facultative.

**Lazy loading (weight: 15)**
In a Web application, data retrieval of other services is sometimes slow. Thus, users may feel that the website is slow if the web page is deployed only after the data is fully processed. Nevertheless, using Lazy Loading allows the page to render and defer only a small portion until the request is ready, pushes the complete content at a later date [27]. This happens in order to provide a better user experience but it also decrease the energy consumption at run-time. Lazy loading is a design model that initialize an object only when it becomes necessary. In this way, Lazy Loading actually contributes to the efficiency of the operation of the software if it is used in a correct and adequate way. This makes Lazy Loading an ideal property for use where network content is accessible and the initialization time has to be reduced to minimum, as in the case of web pages. The inverse of Lazy Loading is Eager Loading [28].

**Object Oriented vs Functional Programming (weight: 10)**
According to our experts, object-oriented programming is lighter and faster than functional programming since it uses only the objects needed to perform the expected functions. This limits the use of material resources to a minimum. As a matter of fact, in object oriented programming, the modeling step is of great importance, since the transcription of real elements in virtual form takes place during this step. In functional programming, all elements are defined as functions and the code runs through

successive calls of functions. Therefore, object-oriented programming is more environment-friendly.

**Cloud Synchronization (weight: 9)**
Terminals we use generate very significant environmental and social impacts, not only in terms of manufacturing, but also in terms of use and end of life. Companies use the cloud to store their data and enable their employers to work remotely. This therefore requires a sharp increase in hardware requirements to back up data, usually accompanied by high expectations in terms of security, which translates into an oversized physical infrastructure. Moreover, data security deployments, the illusion of infinite capability, encourage the unbridled obese computing. Studies begin to appear on the sustainability of cloud use, as [23] that shows that cloud ERP services have a positive impact on the environmental performance of an organization.

**Database VS file (weight: 9)**
The software structure of files is simple. However, the software structure of database is more complicated and robust. Thus, and in order to use less energy in order to preserve the environment, the software structure of the database must be simple and organized in such a way as to facilitate queries.

**Local VS Network Storage (weight: 9)**
This criterion identifies what type of storage is used. Network storage requires a network to store data in data centers. The operation of data centers requires constant energy and cooling, which increases the negative environmental impact. Local storage stores data directly in memory, so it uses only part of the energy needed to operate the computer device. The latter can be turned off, which reduces the negative environmental impact.

**Network Call (weight: 5)**
This criterion indicates the number of requests that the software has submitted. Higher the number of requests, greater the impact of the software on the environment across the hardware and network.

**Compiled VS Interpreted (weight: 5)**
This criterion indicates the type of programming language used: Compiled or Interpreted. Compiled language is faster and translates the code directly into machine language, while interpreted language needs an interpreter, which complicates the procedure and consumes more energy. Thus, compiled language is more favorable in support for environmental requirements.

**Background work (weight: 4)**
This criterion indicates whether the software contains components that work in the background, that is, whether the software works when it is not used by the user. Indeed, such software consumes energy, even if it is in an inactive case. This announces that this criterion makes the software less Green.

**Automatically launch on default startup (weight: 2)**
This criterion means that software dependencies work directly by default at boot time. Therefore, this software consumes energy and hardware components, although they are not used at the user's request.

**Night/Day Mode (weight: 2)**
According to experts, night mode has proven to be less energy-intensive. Thus, its use prolongs the life of the battery, unlike the use of the day mode.

## 4.2 Development sustainability criteria

This phase is composed of 9 criteria.

**Optimize the use of the CPU (weight: 40)**
This criterion indicates the number of accesses to the CPU. The higher the number of accesses to the CPU, the greater the power consumption, and the shorter the life of the CPU. [29] proposes a model for determining a frequency level that minimizes energy consumption during parallel application execution (with an energy saving of 7% for NAS benchmarks).

**Optimizing Algorithms (Human Action) (weight: 14)**
This criterion means that developers have improved already existing algorithms, to improve their performance by making them more sober. The aim is to reduce the use of hardware resources and energy consumption.

**Optimize code instructions ( Compiler action) (weight: 10)**
This criterion determines whether the compiler, by improving the algorithm without the need for developer intervention, makes the software execution procedure greener.

**I/O RAM VS Hard Drive Rate (weight: 10)**
This criterion indicates the number of accesses to the RAM and hard disk. After consulting the experts, it appeared that the hard disk consumes more energy than the RAM, which makes the optimization of the use of the hard disk favorable to the protection of the environment.

**Optimizing the use of the Memory (weight: 5)**
This criterion stresses the importance of the sobriety of the algorithms and the way the software was programmed, in order to preserve the environment. The simpler and more efficient the algorithms and the programming way, the less the software needs access to memory, and the less energy it consumes.

**Binary that takes up space (weight: 5)**
The design should focus on the needs to avoid making the software "obese". The latter will take up a lot of space if it embeds unnecessary code (typical example of the developer copying source code from Stack Overflow). Thus, the software will use more hardware resources with no real benefit to the user. This surplus will apparently take

more binary space than the one imagined in the design phase, and it will consume more of the smartphone's hardware resources without any real benefit for the user. We can say that this way of planning a software with a surplus of features contributes to the programmed obsolescence, and causes the premature renewal of the smartphone.

**Percentage of use of Open Source (weight: 4)**
This criterion defines if the software is developed in Open-source or not. Its advantage lies in the ability of users to use the software without being dependent on the editor and its updates. These updates often render old computer devices obsolete, which requires the purchase of new ones. The ability of users to use, improve, and modify open-source software allows them to extend the life of their computer devices, thereby protecting the environment.

**Bugs (weight: 4)**
This criterion means that the software has a lot of bugs, and if it requires regular maintenance. In this case, the software must be updated regularly. This makes the software more obese, therefore more obsolete.

**The poor legibility of the code to better understand it (Evolution correction) (weight: 4)**
This criterion means that the software is well developed, and that the code is well written and clear. This facilitates development by developers and execution by the compiler.

# 5 Evaluation

In order to validate the matrix, a practical case concerning the "Éco2mix" mobile software, developed by the company RTE, was studied. This application is accessible to everyone, not just RTE customers or its agents. It is integrated into another RTE site, called RTE France. Éco2mix is operable on Android and IOS. This application is dedicated to exposing RTE data on energy uses and production (nuclear, solar, hydraulic, photovoltaic, etc.) both throughout France and at the level of administrative regions and in certain metropolitan areas. The application also gives the average energy consumption of a house in France and the possibility to compare it with those of individuals. Its objective is to better manage the energy balance. Trade in all electrical parameters at the level of the French regions and between France and its neighboring countries has also been included in this application.

## 5.1 Matrix construction

We interviewed the project manager of this application development and filled the sustainability matrix with him (cf. figure 2). Next step is to calculate the score of the application based on the matrix results. Calculus of category score is made on the following way. (a) each level is multiply by its weight. For instance, first criteria obtain the result of (1*30)=30). (b) Then the s um of all results is done for the category and

| Conception Criteria | Weight | 1 | 2 | 3 | 4 | 5 | Explanation |
|---|---|---|---|---|---|---|---|
| Separation between security updates & functional updates | 30 | X | | | | | Éco2mix is often supported by functional updates, and is not always supported by security updates. Eco2mix does not separate the two types of updates. |
| Lazy loading | 15 | | | X | | | Éco2mix interacts with the user based on data already downloaded at the start. Then, as it happens, Éco2mix downloads the necessary data for the operations performed by the user. |
| Object Oriented vs Functional Programming | 10 | | | | X | | The programming used is Object Oriented Programming. |
| Cloud Synchronization | 9 | | | | | X | The cloud is not part of the technical specifications of Éco2mix. |
| Database VS file | 9 | | | X | | | Eco2mix data are organized in both Files and Database. |
| Local VS Network Storage | 9 | | | | | X | Éco2mix stores its data locally, which gives it the right value. |
| Network Call | 5 | | X | | | | Éco2mix does a lot of queries with the network, which decreases the assigned value. |
| Compiled VS Interpreted | 5 | | | X | | | Éco2mix is coded in Java (Compiled), PHP (interpreted) and JavaScript (interpreted). |
| Background work | 4 | | | | X | | Éco2mix does not work on background (except few processes such as the notification process). |
| Automatically launch on default startup | 2 | | X | | | | Éco2mix starts automatically at startup, in order to perform some tasks. |
| Night/Day Mode | 2 | X | | | | | Éco2mix does not support the feature of night/day mode. |
| **Development Criteria** | | | | | | | |
| Optimize the use of the CPU | 40 | | | | X | | Éco2mix does not use the CPU very much. |
| Optimizing Algorithms (Human Action) | 14 | | | | | X | Éco2mix is rather front-end software, so there are not many algorithms to optimize. |
| Optimize code instructions (Compiler action) | 10 | | | X | | | The developers of Éco2mix do not use compiler optimization. |
| I/O RAM VS Hard Drive Rate | 10 | | | | X | | Éco2mix is mobile and web software, it does not use Hard Drive, and it occupies only a small part of the RAM. |
| Optimizing Memory use | 5 | | | | X | | Éco2mix does not use a lot of memory. |
| Binary that takes up space | 5 | | | | | X | Éco2mix is a largely lightweight software, so its binary space is 50MB. |
| Percentage Open Source use | 4 | X | | | | | Éco2mix is developed in closed-source. |
| Bugs | 4 | | X | | | | Éco2mix often suffers from Bugs. |
| The poor legibility of the code to better understand it (Evolution correction) | 4 | | | X | | | The Eco2mix code is a bit complicated. |

**Fig. 2.** Eco2mix sustainability matrix

modified to represent the percentage of sustainability. Here, for the design category, the sum of all weighted criteria is 279, then we calculate its rate over 500 (the maximum score) which give 55,8%, an average score. And the development category gives a rate of 76,875%, which is quite good.

## 5.2 Results evaluation

We discussed these results with the project manager who recognized the results as true for this application. From a sustainability point of view, Éco2mix is a good software both in the design aspect and in the development aspect. Moreover, this interview allowed the project manager to identify some improvement for the future to improve the software sustainability, as follows.

- It would be good for Éco2mix to make the separation between security updates and functional updates and to let the client free to choose the type of update to install.
- Developers have every interest in reducing the number of requests with the network, in order to optimize the environmental performance of Éco2mix.
- The energy performance of Éco2mix is affected by the lack of night/day mode, so it seemed useful to treat this aspect in order to improve the performance.
- It appeared that open-source development is more suitable for the sustainability of software, by facilitating the maintenance, modification and improvement of the code at the level of security as well as functional.
- Developers will be encouraged to optimize the Eco2mix code in order to have fewer bugs, which will increase its environmental performance.
- The code is written in a rather complicated way, which reinforces the interest of making it easier to read, modify and improve.

The project manager found the test useful, as it allowed him to identify the elements that make up sustainability of software, elements that seem relevant to him and that he will use in his next projects.

# 6 Conclusion

We developped a matrix for project managers to test the sustainability of their software. This matrix was based on the results of interviews with experts, as well as a questionnaire published on the web on professional networks. This questionnaire contained two types of criteria: positive and negative making the software more or less durable. We evaluated the sustainability matrix on a smartphone application and discussed the results with the application project manager.

The criteria of the matrix were assembled theoretically on the basis of the results of the interviews and questionnaire, as well as the literature. However, it is preferable for future work to benefit from measurement equipment in order to truly assess the relevance of these criteria and their impact on software sustainability. Each criterion should be tested separately. As mentioned earlier, at this stage there are only sustainability testing tools dedicated to websites. It will therefore be necessary to build

sustainability testing tools for software implemented on all computer devices, such as smartphones and computers.

The matrix forms a good basis for evaluating the durability of software. It has been divided into four categories: Design, Development, Use and Integration. The work presented here covered the first two categories, while the next two categories will be the subject of further interesting work.

## 7 References


1. Munasinghe Institute for Development (MIND). *Integrating sustainable development and climate change in the IPCC fourth assessment report.* Proceedings of the IPCC Expert Meeting (2003)
2. Orgerie A-C. *L'informatique émet plus de gaz à effet de serre que l'aviation.* Pour la Science journal, n°457 (2015)
3. Brauer, B., Eisel, M., Kolbe, L. M. *The State of the Art in Smart City Research – A Literature Analysis on Green IS Solutions to Foster Environmental Sustainability.* Pacific Asia Conference on Information Systems (PACIS). (2015)
4. vom Brocke, J., Loos, P., Seidel, S., Watson, R. T.. *Green IS.* Business & Information Systems Engineering (5:5), pp. 295–297 (2013)
5. Maurey H., Chaize P., Chevrollier G., Houllegatte J-M. *Rapport d'information fait au nom de la commission de l'aménagement du territoire et du développement durable par la mission d'information sur l'empreinte environnementale du numérique.* Sénat. Report N°555 (2020)
6. Byung-Chul C., Hang-Sik S., Su-Yol L, Tak H.. *Life Cycle Assessment of a Personal Computer and its Effective Recycling Rate.* The International Journal of Life Cycle Assessment (2006)
7. Mobbs P., A practical guide to sustainable IT. Commissioned by the Association for Progressive Communications (APC) (2012)
8. Ardito L., Morisio M. Green IT – Available data and guidelines for reducing energy consumption in IT systems Sustainable Computing: Informatics and Systems, Volume 4, Issue 1, Pages 24-32 (2014)
9. Warren-Myers G., The value of sustainability in real estate: a review from a valuation perpective, Journal of Property Investiment and Finance, (2012)
10. Kelly G., Sustainability at home: Policy measures for energy-efficient applicances, Renewable and Sustainable Energy Reviews, Vol 16, Issue 9 (2012).
11. Jones P., Clarke-Hill C., Comfort D., Hillier D., Marketing and Sustainability, Journal of Marketing Intelligence and Planning, vol 26, issue 2 (2008).
12. Dobranskyte-Niskota A., Perrujo A., Jesinghauss J., Jensen P., Indicators to assess sustainability of transport activities, JRC scientific and technical reports, (2009)
13. Dao V., Langella I., Carbo J., From green to sustainability: information technology and an integrated sustainability framework, The journal of strategic information systems, vol 20, issue 1 pp. 63-79 (2011)
14. Loeser F. *Green IT and Green IS: Definition of Constructs and Overview of Current Practices.* 19th Americas Conference on Information Systems (AMCIS) (2013)
15. Murugesan S., *Harnessing Green IT: Principles and Practices.* IT Professional, vol. 10, no. 1, pp. 24-33 (2008)
16. Bordage F. *Empreinte environnementale du numérique mondial.* GreenIT (2019)
17. [greensoft] Naumann, S., Dick, M., Kern, E., Johann, T. The GREENSOFT Model: A reference model for green and sustainable software and its engineering, Sustainable Computing: Informatics and Systems, 1(4), 294–304 (2011)

18. Brauer B., Ebermann C., Hildebrandt B., Remané G., Kolbe L. Green by app : the contribution of mobile applications to environmental sustainability. PACIS 2016, p 220 (2016)
19. Pang C., Hindle A., Adams B., Hassan A. *What Do Programmers Know about Software Energy Consumption?*. IEEE Xplore, Vol 33, Issue 3 (2015)
20. Caroll A., Heiser G. *An Analysis of Power Consumption in a Smartphone*. USENIX conference on USENIX annual technical conference (2010)
21. Deneckère R, Rubio G. EcoSoft: Proposition of an Eco-Label for Software Sustainability. Advanced Information Systems Engineering Workshops. 2020;382:121-132 (2020)
22. Accenture, Cloud Computing and Sustainability: The Environmental Benefits of Moving to the Cloud, Accenture Whitepaper (2010)
23. Gupta S., Meissonier R., Drave, V. A., Roubaud, D., Examining the impact of Cloud ERP on sustainable performance: A dynamic capability view, International Journal of Information Management, Vol51 (2020)
24. Larsson P., Energy-Efficient Software Guidelines, 2011, pp. 14
25. Chang, V and Mills, H and Newhouse, S *From Open Source to long-term sustainability: Review of Business Models and Case studies.* In: Proceedings of the UK e-Science All Hands Meeting 2007. (2007)
26. Ruckebusch P., Giannoulis S., Moerman I., Hoebeke J., De Poorter E., Modelling the energy consumption for over-the-air software updates in LPWAN networks: SigFox, LoRa and IEEE 802.15.4g, Internet of Things,Vol 3–4, Pages 104-119 (2018)
27. del Pilar Salas-Zárate M., Alor-Hernández G., Valencia-García R., Rodríguez-Mazahua L., Rodríguez-González A., López Cuadrado J. L., Analyzing best practices on Web development frameworks: The lift approach, Science of Computer Programming, Vol 102, Pages 1-19 (2015)
28. Fain, Y., Moiseev, A. *Angular Development with TypeScript*, Second Edition. (2018).
29. Sundriyal V., Sosonkina M., Modeling of the CPU frequency to minimize energy consumption in parallel applications, Sustainable Computing: Informatics and Systems, Vol 17, Pages 1-8 (2018)