THE GOOD PRACTICES OF DIGITAL SERVICE ECODESIGN

Authors and date

- Submitted on: Mars 9th 2021
- Benjamin Ninassi; Research engineer ; Inria

INTRODUCTION

Booking a carpool, a train ticket, making a doctor's appointment... these everyday tasks are nowadays easily performed through web platforms. However, these digital services are complex objects, composed of a number of hardware and software layers.

It's a virtual product... but mostly material!

In the case of an online digital service, for example, this materiality can be represented by:

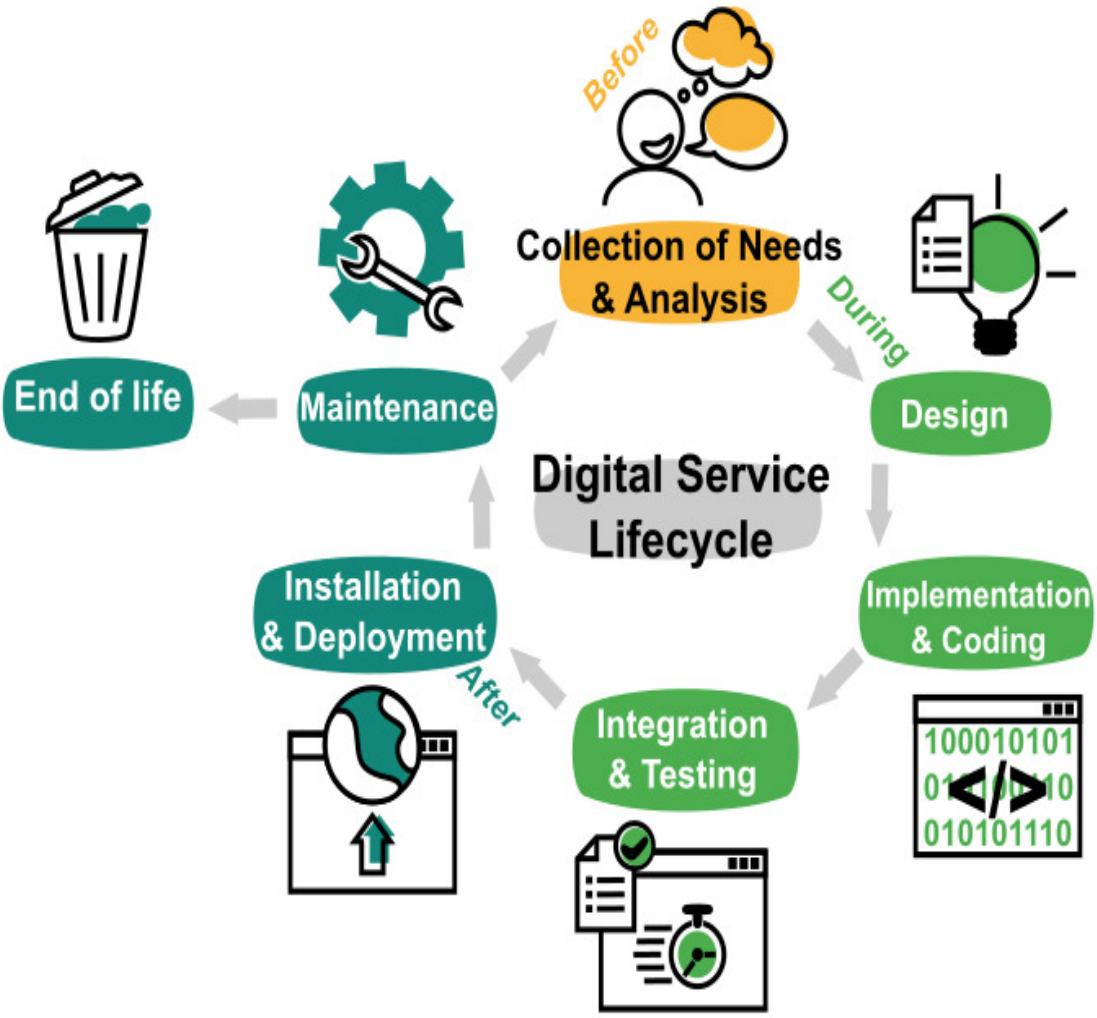
- the servers in the datacenter that host the service
- the end users' devices (computers, phones, tablets...)
- all the network equipments that will transport the data between the servers and the terminals
- other digital services on which it depends

But to give life to a digital service, you need a little more than that!

In this article, we will discuss some important good practices in the eco-design of an online digital service, whether it is a simple website or a more complex web platform, at each stage of its production.

THE CREATIVE PROCESS

As in any project, we can schematize the implementation of a digital service by the three main phases of its life cycle represented below.



Marie Chevallier, 2021

Before

At the beginning of any digital service project, we start by defining the object to be created. What needs do we want to satisfy? Who will be the users? What will they do with it? In short, we define the *What*. Getting precise and relevant answers to these questions is one of the keys to the success of a project, regardless of the technical achievements.

During

The production phase can itself be divided into several parts:

- the design of the software
- implementation
- integration and testing

The **design** aims at answering the question of *How*. It is the definition stage:

- the man-machine interfaces, through mock-ups: *How* will the user interact?
- of the use scenarios: *How* will the user be able to perform the tasks he needs?
- technical architecture and technologies choices: *How* will the developers code?

The **implementation** is the moment when we code, when we implement the necessary technologies. This phase also coincides with the creation of visuals, icons, images, but also with the creation of contents, the collection and formatting of data, which will have to be accessible via the service.

Then comes the **integration** phase, which consists of assembling the different software bricks so that they work together, and inserting the digital service into an existing ecosystem.

Once a first prototype is operational, it will undergo a series of **tests** to validate its quality.

After

After starts with deployment: this is when the digital service becomes operational, most often by deploying it in a datacenter.

This is also when:

- Support, i.e. taking care of users: help with use, answering questions, etc.
- Maintenance, i.e. corrections of anomalies reported by users, developments of security patches, evolutions of functionalities, etc.

When a classic consumer good (like a car for example) starts to be used, its design, its manufacturing and all the other steps that precede its use are finished. This is not the case for an online digital service: the start of a new stage does not end the previous stage. Indeed, during its entire life cycle, an online digital service will evolve:

- by improving or adding new features
- by a succession of technical updates, necessary for security purposes or to correct anomalies
- by deleting features that have become obsolete

CONTINUOUS DEVELOPMENT

Because an online digital service is constantly evolving, the environmental impacts of its production are constantly increasing throughout its life. Thus, each of the steps mentioned above will be repeated for each new version. Most websites have short evolution cycles, from a few weeks to a few months, and the release of a new version is often transparent to its users.

For all these reasons, it is important to minimize not only the environmental impacts of the use of a digital service, but also those of its manufacturing phase and its end of life, thus considering its entire life cycle. This is what we call ecodesign.

A major principle of ecodesign, whatever the field, is to always apply theses two following points in that specific order:

- ***(A) Avoid***: when possible, avoiding creating a new source of impact is always the best choice.
- ***(R) Reduce***: when necessary, seek to reduce impacts as much as possible by showing sobriety.

At each stage of the life cycle of a digital service, ecodesign is possible, here are some examples.

DEFINITION OF NEEDS

Avoid the useless digital service

The service that pollutes the least is the one that doesn't exist. Before you embark on a creative project, make sure the need you want to address is not already covered. Moreover, does this need fit into the Sustainable Development Goals [1]? Will this service have a positive impact on the environment, thus helping to offset its own impacts? Asking the question of usefulness is the first step in ecodesign.

Focus on the primary function and avoid non-essential functionalities

The temptation is great during this phase to want to address a consequent number of needs, to want to reach the largest possible audience, and to ask for features for the sole reason that they exist in other digital services. This type of behavior leads to the creation of unuseable softwares. Several studies (Thomas 2019[2], Cast Software, Standish Group) reveal that a large majority of software features are rarely or never used (between 50% and 80%). Yet they will have had environmental impacts during their production, and their latent presence in softwares also has residual impacts. In a cyclical process, it is also important to question at this stage the relevance of existing features, in order to remove those that have become obsolete.

THE DESIGN

Designing sober interfaces.

Avoid "fat" pages by **reducing** functionality and graphics as much as possible. The service must be usable on a small screen, and if possible *responsive* [3]. In order to reach an optimal relevance, usefulness, fluidity and accessibility must be the priorities of the design. For example, limit the use of images, and when they are essential make sure to **reduce** their size at the source to the minimum necessary.

Think long-term compatibility

It is always better to respect established standards, especially to **avoid** installing a software overlay on the user's terminal, which could be responsible for an induced obsolescence on his hardware. It is for example recommanded to set up a web application rather than a software to install (mobile app or desktop) for an online service. The compatibility of a digital service is built over time by remaining usable on old or exotic devices.

Reuse all or part of existing softwares

Avoid reinventing the wheel. A digital service, especially online, is often an assembly of technologies for which the open source world is very active! It is likely that a software library already exists to do all or part of each feature you want to implement, take advantage of it!

Go for low-tech[4]

Designing tomorrow's digital service that will work on yesterday's hardware not only **reduces** untimely hardware changes but will also be easier to maintain and evolve. A simple SMS can be used to transmit a message, the internet is not the answer to everything.

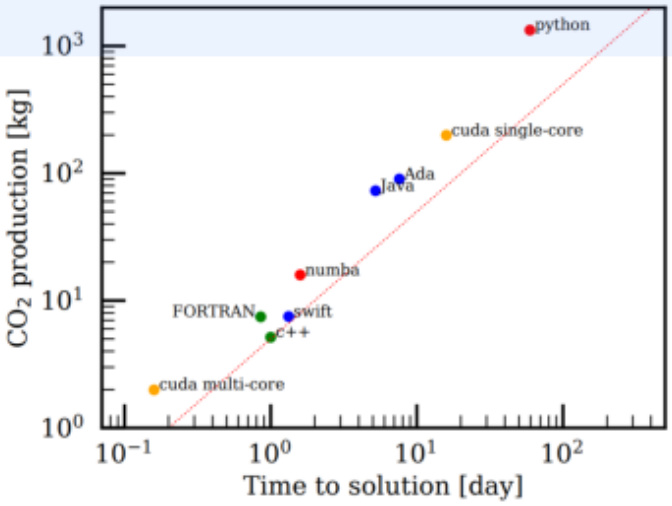
Plan data management

Avoid storing or manipulating unnecessary data. It is tempting to want to store data "just in case". However, this is at best harmful for the environment, at worst illegal under the European Data Protection Regulation [5]. The FAIR[6] principle covers ways to manage data so that they are "Easy to find, Accessible, Interoperable and Reusable". Following this method allows to reduce the impact by setting up a better interoperability and reuse of data.

THE IMPLEMENTATION

Choose a language adapted to the context

Not all languages and technologies are equal, some can be less energy intensive than others. The curve opposite illustrates for example the comparison of the environmental impact of using different languages in astrophysics calculations [7]. This being said, the skills of the developers and the choice of architecture have more impacts than the choice of language: a very good Python developer will optimize his code better than a bad C++ developer, which will ultimately lead to a greater **reduction** of the impacts than the language itself.



Comparison of the environmental impact in CO² equivalent of language choice in astrophysics, Simon P. Zwart

Similarly, reuse of functional code should be favored, regardless of the language. The context, human or technical, must therefore be a determining criterion in the choice of technologies to be implemented.

Choosing sustainable technologies

Technologies implemented in digital services evolve very quickly: new ones appear, and some disappear because of the lack of a strong community to maintain them. Taking an interest in the age and activity of your community allows you to identify sustainable technologies. This is an important criterion to take into account to avoid having to completely rewrite the code of a digital service because one of its technological bricks has become obsolete.

Limit network access

Reduce the traffic generated by your digital service to the bare minimum: you'll reduce its environmental footprint, you'll make it accessible for low-speed connections. The ideal is to design it resilient to disconnections.

Choose simplicity

Avoid complexity. Code that is simple to understand and implement will also be more durable and easier to evolve. This is the basis of the KISS[8] principle: Keep It Simple and Stupid.

Contribute to the creation of digital commons

Avoid others reinventing the wheel: whenever possible, choose an open source license[9] and thus contribute to the world of free software. Not only will you allow others to reduce their own impact, but you may be pleasantly surprised by the community's contributions to your creation!

Measure to Reduce

Use measurement tools to assess the performance of your code. Whether it's the amount of data exchanged, page load time, execution time, number of requests ... everything is subject to optimization. Ecodesign is a never-ending process of continuous improvement. Measurement is essential to this approach in order to know what and how to improve, and to avoid the temptation of the extra functionality or the rebound effect. Nevertheless, finding the right compromise between simplicity and optimization is often a key factor. Beware of the rebound effect: optimization should not be a pretext for adding new non-essential features.

INTEGRATION AND TESTING

Quality for sustainability

The implementation of tests allows to detect anomalies and to ensure an important level of quality of service. This will **avoid** regressions during its evolutions and improve user satisfaction. If your digital service contains too many bugs or instabilities, it may not be used and may be difficult to maintain over time: all the environmental impacts generated by its production will have been generated for nothing. Nevertheless, focus on the most essential tests and beware of their automation, which is itself a source of digital fat.

GOING INTO PRODUCTION

Choosing the right hosting company

Not all datacenters are equal in terms of environmental performances. Choosing a Code of Conduct [10] labeled datacenter is to reduce its impact. Other elements such as its geographical location and its energy efficiency indicator[11] are also to be taken into account.

Size as close as possible to the needs

Choose an infrastructure (how many servers, what type, etc.) as close as possible to the actual needs: trying to anticipate peak loads too much will lead to wasted resources. Prefer the implementation of solutions able to turn on and off the necessary resources on demand, and do not leave unused servers on.

Clean up unnecessary data

Over the life of a digital service, it is common for some of the data it contains to become obsolete. Whether it's old, outdated content or data about past users, it's important to regularly sort out what data is useful and what is no longer useful, so as to **reduce** the impacts of storage.

END OF LIFE

Zombie softwares

Even if a digital service is no longer in use, it will continue to have residual impacts on the environment (due to data storage, hosting, and even regular network accesses to other digital services needed to run it). It is therefore important to think about how to deal with its end of life. In order not to lose the knowledge produced, consider referencing its code in the digital heritage preservation platform *Software Heritage*[12].

SECURING

Security as a resilience factor

From the unavailability of your service to the theft of your users' personal data to the installation of malware on your infrastructure, the consequences of a security breach can be numerous. Corrective actions, when possible, are very costly in time and energy, and therefore impact the environment. The waste is even more important if the service has to be closed following an attack. Security must be a concern at all stages of the life cycle of a digital service, and listing the best practices would require a separate concept sheet. Nevertheless, we can cite as a reference the recommendations of the French National Agency for Information Systems Security [13] and this example from Cloudflare on the treatment of malicious bots [14].

HUMAN IMPACTS

Producing sustainably

Each of these steps will repeat throughout the life of the digital service, and at each of these steps humans collaborate. Through their daily life, these humans will generate impacts: they will use computers, travel to come to their office or to meet, etc. Implementing a corporate environmental policy to limit these impacts is crucial. For example, encouraging telecommuting for employees who have no other solution than driving to the office will **reduce** the carbon emissions related to their travel. Implementing a policy that encourages the extension of the life of computers and the use of refurbished devices will **reduce** the impact of equipment consumption. In general, promote the implementation of an environmental policy in your organization.

Anticipate the transformation of uses

Through its use, your digital service can generate impact by causing a transformation in behavior. For example, an augmented reality application may have been designed to make people want to go hiking and get back in touch with nature, but end up encouraging its users to drive more kilometers to chase virtual chimeras. It is important to be aware as early as possible of the behaviors potentially induced by the use of your future digital service, in order to **reduce** their impacts as much as possible.

These few fundamentals can be completed by:

- Une liste plus exhaustive des bonnes pratiques d'écoconception web[15] par GreenIt
- Une fiche plus complète à destination des développeurs[16] par EcoInfo
- Un référentiel d'écoconception de services numériques piloté par la DINUM, le Ministère de la Transition Ecologique et l'ADEME[17]
- A guide to responsible design of digital services built and made available by the Institut du Numérique Responsable[18]
- Un guide d'écoconception for designers[19] (in french)

Note that these ecodesign best practices are to be applied in addition to all the other more general best practices and methodologies related to the implementation of a digital service, which we will not detail here.

---

1. The Sustainable Development Goals. United Nations ↩
2. Suja Thomas. Feature Adoption Report. Pendo, 2019. Available at the Pendo website ↩
3. Responsive web design (wikipedia) ↩
4. Low technology (wikipedia) ↩
5. Le Règlement Général sur la Protection des Données. Cnil. 2018 ↩
6. Wikipedia: FAIR data) ↩
7. Simon P. Zwart, The Ecological Impact of High-performance Computing in Astrophysics, 2020. Available at ArXiv ↩
8. KISS principle (wikipedia) ↩
9. Open source (wikipedia) ↩
10. Label "Code of Conduct", EU Science Hub ↩
11. Power usage effectiveness (wikipedia) ↩
12. Software Heritage ↩
13. Best Practices, ANSSI ↩
14. John Graham-Cumming. Cleaning up bad bots. Cloudflare, 23/09/2019. Available at The Cloudflare Blog ↩

15. Les 115 bonnes pratiques, Ecoconception web, collectif.greenit.fr ↩
16. Cyrille Bonamy, Cédric Boudinet, Laurent Bourgès, Karin Dassas, Laurent Lefèvre, et al.. Je code : les bonnes pratiques en éco-conception de service numérique à destination des développeurs de logiciels. 2020.(hal-03009741v2) ↩
17. Le Référentiel général d'écoconception de services numériques (RGESN), 2021 ↩
18. Handbook of Sustainable Design of Digital Services, 2021 ↩
19. The intro guide to digital eco-design. Designers Ethiques, 2021 ↩