

# Poster : Bonnes pratiques en éco-conception de service numérique.

## **Cyrille Bonamy**

LEGI – CNRS - UMR5519  
1209-1211 rue de la piscine  
Domaine Universitaire  
38400 Saint Martin d’Hères

## **Cédric Boudinet**

G2Elab – Grenoble INP  
Bâtiment GreEn-ER, 21 avenue des martyrs  
CS 90624  
38031 Grenoble CEDEX 1

## **Laurent Bourgès**

OSUG - Observatoire des Sciences de l'Univers de Grenoble - UAR832  
122 rue de la piscine  
38400 Saint Martin d’Hères

## **Karin Dassas**

CESBIO (Centre d'Etudes Spatiales de la Biosphère)  
UMR5126  
13 Avenue du Colonel Roche  
31400 Toulouse

## **Laurent Lefevre**

Inria, Laboratoire LIP  
Ecole Normale Supérieure de Lyon  
46 allée d’Italie  
69364 Lyon

## **Benjamin Ninassi**

Inria, Rennes  
263 avenue Général Leclerc  
35700 Rennes

## **Francis Vivat**

LATMOS/EcoInfo - CNRS - UMR8190/GDS3524  
11, boulevard d'Alembert  
78280 Guyancourt

## **Résumé**

*Ce poster a pour but de mettre en avant le guide de bonnes pratiques qui a été publié par le Groupement De Service CNRS EcoInfo en 2020 : <https://hal.archives-ouvertes.fr/hal-03009741>.*

*Ce travail est un complément aux trois plaquettes de bonnes pratiques liées au développement logiciel proposées par le réseau des acteurs du DEveloppement LOGiciel au sein de l'Enseignement*

Supérieur et de la Recherche (DevLOG). Celui-ci est plus principalement dédié aux bonnes pratiques en termes d'éco-conception de service numérique qui permettent d'appréhender, de comprendre et de réduire l'impact environnemental du numérique.

Ainsi, ce poster introduit plusieurs concepts à considérer tout au long de la durée de vie d'un logiciel tels que l'effet rebond et les impacts environnementaux des choix de conception et d'utilisation des outils logiciels. La question du lien entre la reproductibilité d'un service numérique et de l'impact environnemental est également abordée, avec notamment l'évocation des plans de gestion de logiciels (« Software Management Plan », SMP, en anglais), des Plans de Gestion des Données (PGD en français, "Data Management Plan", DMP, en anglais) et des principes FAIR (Facile à trouver, Accessible, Interopérable et Réutilisable) pour les données.

L'humain (et notamment les développeurs) dans ce processus d'éco-conception de service numérique est particulièrement important. Il est notamment primordial que les personnels impliqués dans ces développements soient qualifiés et sensibilisés à ces questions environnementales.

Le présent article reprend les pages de la plaquette v3 avec une description introductive des différentes fiches. Des informations supplémentaires sont également données pour préciser en avant-première le contenu de la plaquette v4 qui sera publiée pour les JRES. La fiche Bibliographie, dernière fiche de la plaquette est fournie à la fin, au format de la plaquette.

## **Mots-clefs**

*Eco-conception, service numérique, développement logiciel, effet rebond.*

# 1 Fiches d'introduction

Les quatre premières pages permettent d'introduire la plaquette, de préciser son contexte ainsi que ses raisons d'être (fiche « Mais Pourquoi ? »).

Il est également expliqué que la plaquette suit un fil directeur correspondant au cycle de vie (Avant, Pendant, Après) d'un service numérique, présenté dans la fiche « Quand ? ». Les dernières fiches de la plaquette sont des focus sur des points particuliers, comme le calcul scientifique. La version v4 qui sera présentée aux JRES, dont la publication est prévue pour mars, présentera une fiche « Calcul » plus étoffée, une fiche dédiée « Eco-conception web service », ainsi qu'une fiche « Fin de vie » et une fiche « Rupture ».

## 1.1 Introduction et contexte

**Je code : les bonnes pratiques en écoconception de service numérique à destination des développeurs de logiciels**

**Auteurs :**  
Cyrille Bonamy : LEGI / CNRS  
Cédric Boudinet : G2Elab / Grenoble INP  
Laurent Bourghès : OSUG / CNRS  
Karin Dassas : IAS / CNRS  
Laurent Lefèvre : Inria / ENS Lyon  
Francis Vivat : LATMOS / CNRS

Les auteurs sont membres du [Groupe de Service CNRS EcoInfo \[1.1\]](#) qui travaille sur l'écoresponsabilité du numérique.

**Résumé :**

Cette plaquette est un complément aux 3 plaquettes de bonnes pratiques liées au développement logiciel proposées par le réseau des acteurs du Développement LOGiciel au sein de l'Enseignement Supérieur et de la Recherche : DevLOG.

Ce volet est dédié aux bonnes pratiques en termes d'écoconception de service numérique qui permettent d'appréhender, de comprendre et de réduire l'impact environnemental du numérique.

Après avoir explicité le contexte général dans une première fiche, une seconde fiche ("Mais pourquoi ?") met en évidence la nécessité d'intégrer une dimension environnementale dans nos conceptions de service numérique, et par conséquent dans nos développements de logiciels. La troisième fiche ("Quand ?") rappelle les étapes du cycle de vie d'un service numérique pour introduire les fiches de bonnes pratiques qui correspondent aux différentes étapes : "Avant", "Pendant" et "Après", en gardant à l'esprit que le développement est souvent itératif, et les frontières entre les différentes étapes sont perméables.

Vous trouverez à la fin de la plaquette une fiche spécifique sur les bonnes pratiques d'écoconception pour le calcul scientifique, ainsi que des fiches sur le développement sur plateforme mobile, pour le web et sur accélérateur.

Les plaquettes DevLOG existantes :

- [Je code : les bonnes pratiques de développement logiciel \[1.2\]](#)
- [Je code : les bonnes pratiques de diffusion \[1.3\]](#)
- [Je code : quels sont mes droits et obligations ? \[1.4\]](#)

**Remerciements** pour leur lecture à : Françoise Berthoud, Rémi Cailletaud, Christophe Cossou, Loïc Maurin, Gabriel Moreau, Patrick Moreau, Olivier Ridoux, Jean-Christophe Souplet

**Un service numérique, c'est :**

- de l'information : les données
- des traitements : algorithme, filtrage, simulation
- des échanges d'informations
- des interfaces utilisateurs

**Un service numérique repose donc sur :**

- des infrastructures logicielles : applications, outils, bibliothèques, protocoles
- des infrastructures matérielles : serveurs, équipements réseau, terminaux, capteurs
- des personnes : développeurs, administrateurs systèmes et réseaux, chefs de projet, chercheurs

**Exemple de service numérique : le workflow d'une simulation numérique.**

Il est constitué des briques suivantes :

- la préparation des données d'entrée
- le transfert des données d'entrée vers la plateforme de calcul considéré (machine locale, mésocentre régional, [GENCI \[2.1\]](#), cloud public, GPU/CPU)
- le calcul sur la plateforme de calcul
- le transfert des données vers la plateforme de post-traitement
- le post-traitement, stockage et dissémination des données de sortie du calcul
- l'analyse et l'exploitation des données

De nombreux leviers pour réduire les impacts, mais une **vision globale** est nécessaire :

- limiter les transferts de données
- choix plateforme de calcul et de pré et post-traitement
- post-traiter les données au plus proche du lieu de création
- limiter les données d'entrée et/ou de sortie
- choisir des briques logicielles externes ou à redévelopper en interne

**Eco-concevoir un service numérique, c'est intégrer les aspects environnementaux tout au long de son cycle de vie [2.2] (attention à bien définir l'unité fonctionnelle [2.3])**

*Développeurs, mais aussi chefs de projet, ingénieurs et chercheurs, tous et toutes sont concernés par cette plaquette !*

V3 20/05/2021 CC-BY-SA-4.0

Figure 1: Plaquette Fiches Introductives

## 1.2 Pourquoi et Quand ?

La fiche « Pourquoi » montre que l'impact environnemental du numérique est loin d'être négligeable. D'où la nécessité d'essayer de le réduire à chaque étape du cycle de vie d'un service numérique.

Dans la v4, l'écosystème « green IT » sera mis à jour dans la fiche « Pourquoi ».

Les actions (et fiches correspondantes) « Avant », en jaune orangé, correspondent à la collecte et à l'analyse des besoins en amont du développement (phase de conception incluse). Elles sont suivies des phases « Pendant », en bleu, puis des phases « Après », en vert, qui ont lieu après le développement du service numérique. Les phases peuvent être itératives.

Une flèche « Fin de vie » sera rajoutée en marge du cycle de la fiche Quand.

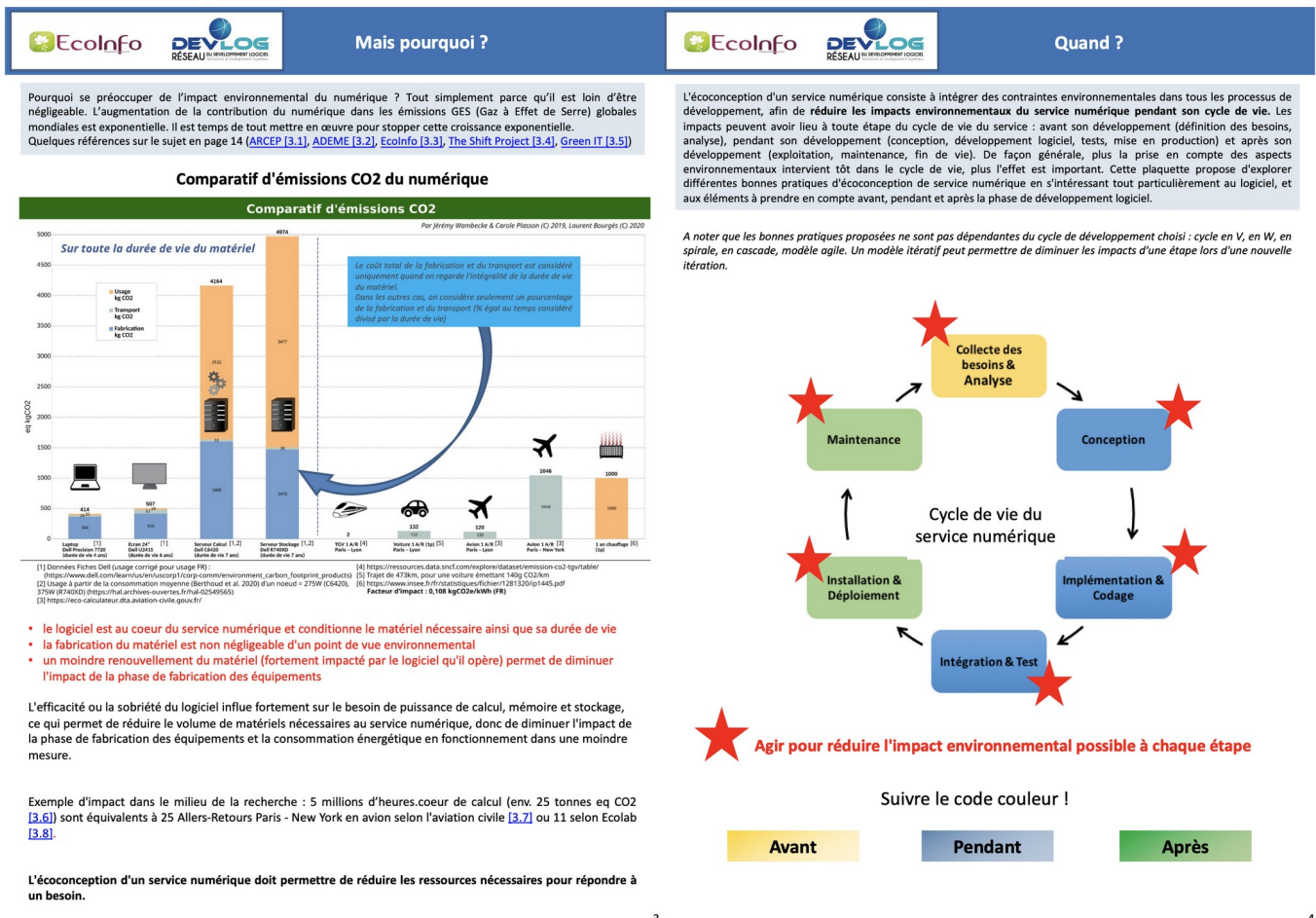


Figure 2: Plaquette Pourquoi Quand

## 2 Phase Avant

Les trois fiches « Avant » (code couleur orange pour l'en-tête) couvrent cette phase.

Avant de se lancer dans le développement du service numérique, il est important de se poser certaines questions. De quelles fonctions a-t-on vraiment besoin ? Est-t-il possible de réutiliser des briques existantes, de favoriser le logiciel libre ? Quelles infrastructures et quels langages doit-on utiliser, en fonction des besoins et aussi de leurs impacts environnementaux ? C'est également le moment de planifier la gestion du logiciel (SMP) et des données (DMP).

Avant

S'appuyer sur les grands principes d'écoconception de service numérique :

- simplicité** : simplifier le logiciel pour éviter les usines à gaz
  - en termes de fonctionnalités : 70 % des fonctionnalités demandées par les utilisateurs ne sont jamais ou rarement utilisées (Standish Group, 2006)
  - en termes d'interfaces utilisateurs
- frugalité et sobriété** : limiter le nombre et la taille des éléments (images par exemple). Par exemple, dans un développement web, éviter les pages "obèses" en terme de fonctionnalités et de graphisme.
- pertinence** = utilité (le résultat doit répondre à l'attente de l'utilisateur) x rapidité (temps de réponse pour l'utilisateur) x accessibilité (par exemple pour certains handicaps)
- durabilité** : réutiliser tout ou partie d'un logiciel permet d'éviter de dupliquer les développements; contribuer pour le bénéfice de la communauté.

Je maîtrise le nombre de fonctionnalités logicielles : éviter l'obésiciel

**Trop de fonctionnalités disparates** : où est passée la fonction que j'utilisais le plus souvent ? Mon logiciel est devenu un obésiciel ! Mon service numérique devient une véritable "usine à gaz" !

**Plus de fonctionnalités que nécessaire** : mon ancienne infrastructure ne suffit plus, je dois en changer !

**Je favorise le libre : réutiliser des briques logicielles et contribuer aux communs**

**Je planifie la gestion du logiciel : accroître la durée de vie**

Un plan de gestion logiciel (**SMP** Software Management Plan) est un outil pour la pérennisation du logiciel. Les informations sur le logiciel, sur son contexte, sur ses caractéristiques, ou sur l'organisation de sa diffusion sont ainsi regroupées, mises à jour au cours du cycle de vie, et permettent d'en faire un produit modifiable et réutilisable facilement. [Opidor \[5.2\]](#), [Presoft \[5.3\]](#) (voir aussi le [Data Management Plan \[5.4\]](#) Slide 7)

Avant

Prendre le temps de corréliser le choix d'un langage et de la plateforme de déploiement tout en considérant les contraintes globales du service numérique complet.

Je réfléchis au déploiement du service : s'adapter au mieux au contexte

**En fonction des caractéristiques des plateformes disponibles :**

- microcontrôleur ou processeur embarqué
- ordinateur ou serveur local
- plateforme spécialisée : cluster spécialisé HPC, GPU, FPGA, ARM64
- Offre De Service (ODS) de site ou de tutelles (CNRS, Universités)
- clouds publics
- lieu géographique de la plateforme
- capacités disponibles

**En fonction des contraintes du service :**

- langages supportés, communications spécialisées (Infiniband, OmniPath, SpaceWire, Série...)
- goulots d'étranglement : accès disque, réseau, transferts mémoire
- pérennité, portabilité, sécurité, coûts à long terme, maintenance, temps de retour
- lieu géographique des usages, tutelles commanditaires du service

**Sans oublier les contraintes environnementales**

**Je choisis mon langage et/ou ma pile logicielle : tout est affaire de compromis**

Illustrations de l'efficacité énergétique en fonction du langage :

- à gauche : [Simon P. Zwart, Nature Astronomy \[6.1\]](#) avec une simulation N-Body, figure actualisée par [P. Augier \[6.4\]](#) et [\[6.5\]](#) mettant en évidence l'importance des compilateurs Python et des facteurs humains (optimisation du code, expertise et temps passé pour optimiser, tester et mesurer les codes),
- à droite, [Rui Pereira et al, SLE17 \[6.2\]](#) [\[6.3\]](#) basé sur le Computer Language Benchmark Game.

Time & Memory	Energy & Time	Energy & Memory	Energy & Time & Memory
C++ Intel-Clang	C	C# Visual	C# Pascal + C++
Fortran + C++ + Fortran	Fortran	Fortran + C++ + Fortran + Go	Fortran + C++ + Fortran
Ada	Ada	Ada	Ada
Java + Chapel + Lisp + Occam	Java	Java + Chapel + Lisp	Java + Chapel + Lisp + Occam
Haskell + C++	Haskell + C++	Haskell + C++	Haskell + C++
Swift + PHP	Swift + Chapel	C# + PHP	Dart + F# + Backtick + Haskell + PHP
F# + Backtick + Haskell + Python	Lisp + Occam + Go	Dart + F# + Backtick + Haskell + Python	JavaScript + Ruby + Fortran
JavaScript + Ruby	Fortran + Haskell + C++	JavaScript + Ruby	TypeScript + Erlang
Dart + TypeScript + Erlang	Swift	Dart + F#	Erlang + Lua + Perl
Ruby + Perl	JavaScript	Ruby	
Lua	TypeScript + Haskell		
	PHP		
	Erlang		
	Lua + Ruby		
	Ruby		

L'usage de Python illustre le plus grand écart (~100x) en efficacité selon l'environnement d'exécution (python, numba, pythran, pypy) et l'optimisation du code dans ce cas d'utilisation (performance-driven development).



- langages compilés (natifs ou habituellement interprétés mais optimisés, p. ex. grâce à Numba ou PyThran pour Python) à privilégier pour les traitements lourds, haute performance ou temps réel
- langages faciles d'accès (interprétés) à privilégier pour les traitements moins contraints, afin de faciliter la maintenance, le ré-usage et ainsi la durabilité
- pour les langages interprétés, les performances peuvent être grandement améliorées par l'utilisation de libraires compilées
- toutes les briques ne doivent pas forcément être écrites dans le même langage

Figure 3: Plaquette Fiches Avant

JRES 2021 – Marseille

5/10

La troisième fiche « Avant » est entièrement dédiée aux données. Les rendre interopérables et facilement accessibles permet de limiter leurs impacts environnementaux.

Avant

Anticiper, prendre en compte les données pour diminuer leur impact environnemental. Tenir compte de la volumétrie et des coûts de transfert des données. Favoriser l'interopérabilité et les données ouvertes.

**Je fais attention à la volumétrie des données : sobriété numérique**

**Stockage des données : ASCII ou formats binaires ?**

Le format texte (ASCII brut, XML, YAML...) nécessite généralement plus de place pour stocker la même information. Par contre lorsqu'on versionne les fichiers, le phénomène peut s'inverser (stockages successifs vs. stockage différentiel). Il y a un risque d'obsolescence à l'utilisation des formats binaires non ouverts.


**Volumétrie importante ?**

- je privilégie le traitement au plus près des données (serveur) pour éviter de transférer les jeux de données sur les postes des utilisateurs (et besoin d'autres machines puissantes)
- je tiens compte de la volumétrie des données manipulées pour concevoir les traitements et éviter les transferts volumineux entre les couches logicielles
- je minimise la volumétrie du paquet logiciel (archive) et j'élimine les dépendances superflues.

**Je planifie la gestion des données : durabilité, diminution des développements redondants**


F

Findable




A

Accessible




I

Interopérable



R

Reusable



source :SangyaPundir CC BY-SA 4.0

Suivre le principe FAIR permet de réduire l'empreinte environnementale pour plusieurs raisons :

- **Interopérabilité** : les données doivent être faciles à combiner avec d'autres jeux de données, à la fois par les humains et par les systèmes informatiques. Choisir de rendre les données interopérables permet d'éviter de multiplier des données avec des formats différents : diminution du nombre de données et du coût de conversion.
- **Open Data / Reproductibilité** : rendre les données et les codes sources facilement accessibles et reproductibles permet de réduire l'empreinte environnementale en évitant des doublons. Moins de stockage (utilisation de web-services), moins de développement redondant.

De nombreuses ressources existent : [principes FAIR \[7.1\]](#) / [directive européenne \[7.2\]](#) / [European Open Science Cloud \[7.3\]](#) / [Research Data Alliance \[7.4\]](#)

Les objectifs souvent cités d'un [plan de gestion de données \(DMP\) \[7.5\]](#) de la recherche sont les suivants : assurer la reproductibilité des données, améliorer l'impact des projets de recherche et leur contribution scientifique, et même satisfaire les exigences de financeurs.

Un des autres objectifs de ce plan est environnemental : un plan correctement fait permet d'aboutir à des données FAIR. Exemples de modèles de DMP sur le site [OPIDOR \[7.6\]](#)

Voir aussi l'[Atelier des données \[7.7\]](#)

Figure 4: Plaquette Fiche Avant Data

### 3 Phase Pendant

Les deux fiches « Pendant » (code couleur Bleu) couvrent cette phase.

Optimiser son logiciel, comme implémenter des algorithmes efficaces et les structures de données adéquates, permet de réduire le temps d'exécution, ce qui conduit en général à une réduction de la consommation électrique, et donc à une réduction des émissions eqCO2. Mais attention à l'effet rebond, et aux effets sur les autres éléments du service numérique.

De même il faut être vigilant lorsque l'on applique les bonnes pratiques usuelles en ingénierie logicielle (gestion de version, intégration continue). Il faut les appliquer, mais avec des précautions (oui, mais...).

  <span style="float: right;"><b>Pendant</b></span>	  <span style="float: right;"><b>Pendant</b></span>
<p>Analyser et superviser mon service numérique afin d'identifier les briques logicielles et les fonctions les plus consommatrices.</p>	<p>Optimiser son logiciel, comme implémenter des algorithmes efficaces et les structures de données adéquates, permet de réduire le temps d'exécution, ce qui conduit en général à une réduction de la consommation électrique, et donc à une réduction des émissions eqCO2. Mais attention à l'effet rebond, et aux effets sur les autres éléments du service numérique. De même il faut être vigilant lorsque l'on applique les bonnes pratiques usuelles en ingénierie logicielle (gestion de version, intégration continue). Il faut les appliquer, mais avec des précautions (oui, mais...).</p>
<p><b>J'analyse mon code : identifier a priori</b></p> <ul style="list-style-type: none"> <li>• les analyses statiques permettent, entre autres, de déterminer la difficulté de maintenance d'un logiciel (exemples d'outils : <a href="#">Sonarube</a>, <a href="#">codacy</a>)</li> <li>• les analyses dynamiques permettent de quantifier l'usage des ressources (exemples outils : <a href="#">gcov</a>, <a href="#">gprof</a>, <a href="#">perf</a>, <a href="#">valgrind</a>)</li> </ul> <p>Ces analyses de logiciel - qui font partie des bonnes pratiques classiques - sont indispensables afin de réduire l'impact d'un service numérique.</p>	<p><b>J'optimise mon code (oui mais...)</b></p> <p><b>Attention à l'effet rebond :</b> </p> <p>Optimiser un logiciel peut induire à lancer davantage d'opérations ou traiter davantage de données, donc l'empreinte écologique du service ne sera pas réduite (Paradoxe de Jevons).</p> <p>L'optimisation devrait servir simplement à réduire la consommation énergétique et des ressources, et si possible d'arriver plus vite au résultat. Chaque exécution a un impact !</p> <p>Il est primordial de n'optimiser que ce qui a le plus d'impact (Loi de Pareto).</p> <p>Il ne faut pour autant pas négliger la qualité des tests dans le processus d'optimisation (régressions).</p>
<p><b>Je mesure les performances : identifier en fonctionnement</b></p> <ul style="list-style-type: none"> <li>• temps de chargement</li> <li>• temps d'exécution</li> <li>• consommation électrique</li> <li>• consommation des protocoles réseau (par exemple dans l'IOT)</li> <li>• taille des données</li> <li>• nombre de requêtes</li> <li>• performance web</li> </ul> <p>Exemples d'outils : <a href="#">Firefox web tools</a> (trafic, requêtes, JavaScript), <a href="#">Apache JMeter</a> (scénarios web), <a href="#">ecoindex</a> (web), profiler intégré et adapté au langage.</p>	<p><b>J'améliore mon code (oui mais...)</b></p> <p>L'amélioration est un problème multifactoriel : architecture, utilisation des ressources processeur, mémoire, stockage et trafic réseau. L'utilisation d'un levier d'amélioration peut nuire à l'usage d'une autre ressource.</p> <p>Par exemple :</p> <ul style="list-style-type: none"> <li>• adopter un système de cache permet d'accélérer les traitements, mais attention aux impacts sur la mémoire, le stockage et la complexité accrue du logiciel</li> <li>• utiliser la compression pour réduire le trafic réseau augmente légèrement l'utilisation du CPU</li> <li>• sécuriser peut augmenter le coût énergétique qui croît avec la complexité des algorithmes de sécurisation. Certains algorithmes sont déjà câblés dans les processeurs (p.ex. AES), il faut les utiliser ! Mais l'absence de sécurisation peut engendrer d'autres coûts (maintenance accrue, remise en service ...) <a href="#">[9.1]</a></li> </ul>
<p><b>Un focus : je profile la consommation énergétique des logiciels</b></p> <p>Chaque usage de ressource (processeur, mémoire, espace de stockage, réseau) dans mon logiciel provoque une consommation électrique.</p> <p>Pour mesurer l'impact énergétique des logiciels en phase d'usage : besoin d'équipements matériels (PDU, wattmètres) ou de sondes logicielles.</p> <p>Je profile la consommation électrique dans le temps, pour découvrir les différentes phases intensives de mon logiciel.</p>  <p style="text-align: right; font-size: small;">Wattmètre WattsUp Pro - PDU Eaton</p> <p style="text-align: right; font-size: x-small;">source : Laurent Lefèvre Inria</p>	<div style="display: flex;"> <div style="flex: 1;"> <p><b>Gestion de version (oui mais...)</b></p> <p>J'utilise un outil de gestion de version, mais :</p> <ul style="list-style-type: none"> <li>• j'évite ou limite d'y stocker les paquets binaires et les jeux de données non indispensables</li> <li>• je ne place pas en gestion de version les produits de compilation ni les fichiers de sortie</li> </ul> </div> <div style="flex: 1;"> <p><b>Intégration continue (oui mais...)</b></p> <ul style="list-style-type: none"> <li>• je réfléchis à mon Intégration Continue (CI). Je choisis un docker de taille minimum, et j'active ma CI uniquement sur certaines branches et j'envisage une exécution programmée. Ainsi je n'exécute pas tous les tests et ne produis pas tous les fichiers à chaque modification</li> <li>• je surveille la durée des jobs, leur nombre, la taille des artefacts, le trafic réseau</li> <li>• je privilégie les forges mutualisées</li> </ul> </div> </div>

Figure 5: Plaquette Fiches Pendant

## 4 Phase Après

Deux fiches « Après » (code couleur Vert) couvrent cette phase dans la v3. Une troisième fiche « Fin de vie » (gestion des déchets notamment) sera disponible dans la v4.

L'écoconception d'un service numérique doit prendre en compte les impacts liés à la distribution, l'utilisation et l'administration du logiciel, sans oublier les besoins des utilisateurs. Il faut s'assurer que l'efficacité énergétique et la réduction des impacts environnementaux sont conservées avec les évolutions du logiciel. Sensibiliser les utilisateurs pour améliorer les usages est aussi un point important.

Les fiches « Après » déjà existantes seront également mises à jour dans la v4.

**Après**

L'écoconception d'un service numérique doit prendre en compte les impacts liés au déploiement et à la production. L'amélioration continue permet de réduire les impacts environnementaux.

**Déploiement : sobriété numérique**

- je privilégie un hébergement mutualisé, labélisé COC (Code Of Conduct), au plus près des données et des utilisateurs
- je privilégie la virtualisation (moins de serveurs physiques), en minimisant l'empreinte mémoire et disque (taille des machines virtuelles, des containers), sauf cas particuliers (calcul haute performance)
- je fais attention à certains effets rebond : j'évite la multiplication des machines virtuelles, des instances du service

**Production : amélioration continue en fonction des usages**

- j'exploite la supervision (et les alertes) pour observer les pics CPU, ressources utilisées (disque, réseau), consommation électrique
- je modifie le service numérique pour l'adapter en fonction de l'usage observé (amélioration continue)
- je réduis les fréquences et volumes des sauvegardes, j'adopte la déduplication

Exemples d'outils de supervision : top, vmstat, zabbix, scalasca, nagios, prometheus, grafana

Exemples d'outils utilisés pour l'amélioration continue du service numérique  
(source : PNGEgg, adaptée par C. Bonamy)

**Après**

L'écoconception d'un service numérique doit prendre en compte les impacts liés à la distribution, l'utilisation et l'administration du logiciel, sans oublier les besoins des utilisateurs. Il faut s'assurer que l'efficacité énergétique et la réduction des impacts environnementaux sont conservées avec les évolutions du logiciel.

**Je traque les ressources inutiles : éviter le gaspillage énergétique**

**La consommation statique des machines est importante : j'éteins les machines si possible**

Le logiciel influence la consommation dynamique des machines, mais il y a peu de proportionnalité énergétique (un serveur inoccupé peut consommer 50 % de son budget électrique). La consommation statique d'un serveur ou d'un équipement réseau est importante

Luiz André Barroso and Urs Hölzl, "The case for Energy-Proportional Computing", IEEE Computer, 2007 111-11

**Mise en place de systèmes de mise en veille**

- j'éteins les machines virtuelles qui ont une empreinte mémoire et CPU
- je limite le nombre de services déployés
- je favorise les extinctions d'hôtes et de ressources (cœurs de calcul, systèmes de stockage) inutilisés

**Je distribue et maintiens mon code : favoriser la durabilité et la simplicité**

**Diffusion**

- je dépose le logiciel en un endroit unique et facilement accessible
- déclaration auprès de [Software Heritage \[11.2\]](#)

**Gestion des mises à jour**

- je réduis la taille des produits logiciels
- je rationalise leur nombre et leur fréquence
- rétrocompatibilité autant que possible
- je veille à éviter l'ajout constant de fonctionnalités
- workflow des mises à jour clairement défini
- [Long Term Support \[11.3\]](#) pour les mises à jour correctives

**Je sensibilise : améliorer les usages**

- en m'appuyant sur ce qui existe :
  - [EcoInfo \[11.4\]](#)
  - [ADEME \[11.5\]](#)
  - [principles green \[11.6\]](#)
- en affichant des informations de suivi de consommation
- je sensibilise les utilisateurs de mon service numérique sur les impacts environnementaux de leurs usages

Figure 6: Plaquette Fiches Après



## 5 Fiches spécifiques

Les deux dernières fiches sont des focus sur certains services numériques spécifiques, ou des contextes spécifiques.

La fiche « Calcul Scientifique » sera sur deux pages dans la v4, avec un volet IA.

Une fiche dédiée à l'écoconception Web Service sur une page entière remplacera l'encart existant sur la fiche Autres fiches.

Il est aussi prévu de rajouter un encart ou une fiche « Rupture », qui évoquera des mouvements plus radicaux comme CollapseOS.





  <b>Fiche Calcul Scientifique</b>	  <b>Autres Fiches</b>
<p><b>J'éco-conçois du code scientifique</b></p> <p><b>Avant de développer :</b></p> <ul style="list-style-type: none"> <li>je choisis la licence (si possible ouverte afin de faciliter le ré-usage et la reproductibilité)</li> <li>je choisis la technologie cible et le langage en fonction des contraintes (C++, Python, Fortran, GPU, Cluster classique, ARM, Cloud, Web services)</li> <li>je choisis l'infrastructure adaptée pour chaque phase (laboratoire pour développement, méso-centre pour mise au point et validation, centre de calcul national pour production)</li> <li>plus généralement, je conçois l'ensemble du workflow (données d'entrée, traitement, post-traitement, stockage) en pensant aux impacts environnementaux (coût des transferts réseaux...)</li> <li>j'alerte les commanditaires et autres développeurs des impacts sur l'environnement du numérique (en m'appuyant sur les instances existantes : EcoInfo)</li> </ul> <p><b>Au tout début du développement :</b></p> <ul style="list-style-type: none"> <li>je crée un dépôt (git via un gitlab universitaire par exemple)</li> <li>je crée les fichiers "AUTHORS", "LICENSE", "README"</li> <li>je mets en place l'intégration continue (Gitlab pipeline, Travis-CL...), sans excès inutiles</li> <li>je prévois la création de la documentation et sa mise à jour (Sphinx, Doxygen...)</li> <li>j'envisage les éventuels plantages et prévois la possibilité de redémarrage de mes calculs</li> </ul> <p><b>Pendant le développement :</b></p> <ul style="list-style-type: none"> <li>je diffuse les sources, partage le dépôt, dès que possible</li> <li>j'assure la reproductibilité de mes expériences numériques</li> </ul> <p><b>Après, pendant les phases de production :</b></p> <ul style="list-style-type: none"> <li>je recherche les points chauds, à l'aide d'outils existants (gprof, valgrind...)</li> <li>je fais attention au poids des I/O (entrées/sorties)</li> <li>j'analyse la scalabilité de mon code (idéalement sur l'infrastructure d'utilisation finale)</li> <li>je me fais aider pour optimiser (exemple : <a href="#">Performance Optimisation and Productivity [12.1]</a>) tout en faisant attention aux effets rebond (toujours plus!)</li> <li>je sensibilise les utilisateurs de mon code sur les impacts environnementaux de leurs calculs</li> <li>je recherche le meilleur compromis entre temps de retour, finesse du calcul et coût eqCO2</li> <li>je réfléchis au réel intérêt de mes calculs et je n'hésite pas à redévelopper si besoin</li> </ul> <p><b>Quelques liens intéressants :</b></p> <ul style="list-style-type: none"> <li><a href="#">Loi d'Amdahl [12.2]</a></li> <li><a href="#">MPI [12.3]</a>, <a href="#">OpenMP [12.4]</a>, <a href="#">OpenACC [12.5]</a>, <a href="#">OpenCL [12.6]</a></li> <li><a href="#">Comment obtenir des ressources (calcul et stockage) dans l'ESR? [12.7]</a></li> </ul> <p><b>Je ne pense pas :</b> "c'est un petit code, ça ne servira qu'à moi, ces bonnes pratiques ne me sont pas destinées".</p>	<p><b>Comment éco-concevoir des plateformes mobiles et objets connectés ?</b></p> <ul style="list-style-type: none"> <li>spécifique aux objets connectés : utiliser le mode Deep-Sleep (*) entre chaque mesures</li> <li>le codage des nombres a un impact direct sur les performances et donc la consommation. Choisir la bonne représentation (virgule flottante vs fixe) et la bonne taille permet de diminuer la fréquence processeur, voire de downsizer le processeur</li> <li>les objets connectés échangent beaucoup de données par internet, à volume de données finales échangées identique certains protocoles ajoutent plus ou moins de données superflues (ex. MQTT, HTTP, XMPP)</li> <li>l'architecture logicielle (interruptions vs. attente active, buffers circulaires, optimisation) a un impact direct sur la charge processeur</li> </ul> <p>(*) Deep-Sleep : mode particulier dans lequel le microprocesseur ne consomme plus que quelques <math>\mu A</math> et peut être réveillé par une interruption</p> <p><b>Comment éco-concevoir mon développement Web ?</b></p> <p>Je vais faire un tour sur ce site : <a href="#">Les 115 bonnes pratiques [13.1]</a></p> <p>S'il fallait en retenir trois :</p> <ul style="list-style-type: none"> <li>minimiser le nombre de requêtes et le volume de données</li> <li>réduire l'impact du javascript (poste client)</li> <li>réduire la taille et le nombre des ressources web (images, styles, scripts, documents ...)</li> </ul> <p>Je teste mon site web avec <a href="#">Ecoindex [13.2]</a> et <a href="#">Ecometer [13.3]</a></p> <p>Autres outils de mesure : FireFox web tools (traffic, requetes, JS), Apache JMeter (scénarios web) (pour l'amélioration continue).</p> <p><b>J'éco-conçois du code sur un accélérateur (GPU)</b></p> <ul style="list-style-type: none"> <li>je fais du code portable et performant (attention à trop de spécificités avec un type d'accélérateur, comme par exemple CUDA)</li> <li>je choisis la technologie et les contraintes associées en fonction de mes besoins (Cuda vs OpenCL : performance et simplicité vs généricité et verbosité)</li> <li>la clé des accélérateurs est de réfléchir à uniformiser (pas de conditions if/else), pas d'hétérogénéité entre les cœurs) et maximiser les traitements sur l'accélérateur (GPU) afin d'éviter les transferts mémoire CPU vers GPU (attention au temps de transferts mémoire CPU/GPU vs kernel time)</li> <li>je suis vigilant à l'impact des entrées/sorties (I/O) qui peut fortement dépendre de l'infrastructure cible (poste de travail vs cluster spécialisé)</li> <li>je fais attention au poids de l'apprentissage pour l'Intelligence Artificielle</li> <li>je sensibilise les utilisateurs au coût des calculs, et notamment au coût global (qui comprend la phase d'apprentissage)</li> </ul>

Figure 7: Plaquette Fiches Spécifiques

## 6 Bibliographie

La Bibliographie est celle de la plaquette v3. Son format a été gardé car il permet de faire référence aux différentes pages de la plaquette.



**EcoInfo** **DEVLOG**  
RÉSEAU EN DÉVELOPPEMENT ÉCOLOGIQUE

### Références

- [1.1] <https://ecoinfo.cnrs.fr>
- [1.2] [https://hal.archives-ouvertes.fr/hal-02083801/file/20191202\\_plaquette\\_developpement\\_V1.1.pdf](https://hal.archives-ouvertes.fr/hal-02083801/file/20191202_plaquette_developpement_V1.1.pdf)
- [1.3] [https://hal.archives-ouvertes.fr/hal-02400300/file/20191202\\_plaquette\\_diffusion\\_V1.1.pdf](https://hal.archives-ouvertes.fr/hal-02400300/file/20191202_plaquette_diffusion_V1.1.pdf)
- [1.4] [https://hal.archives-ouvertes.fr/hal-02399517/file/20191202\\_plaquette\\_pl\\_licences\\_V1.1.pdf](https://hal.archives-ouvertes.fr/hal-02399517/file/20191202_plaquette_pl_licences_V1.1.pdf)
  
- [2.1] <https://www.genci.fr>
- [2.2] <https://www.eco-conception.fr/static/analyse-du-cycle-de-vie-acv.html>
- [2.2] <https://www.eco-conception.fr/static/fonction-unite-fonctionnelle-acv.html>
  
- [3.1] [https://www.arcep.fr/uploads/tx\\_gspublication/reseaux-du-futur-empreinte-carbone-numerique-juillet2019.pdf](https://www.arcep.fr/uploads/tx_gspublication/reseaux-du-futur-empreinte-carbone-numerique-juillet2019.pdf)
- [3.2] [https://www.institutparisregion.fr/fileadmin/NewEtudes/Etude\\_1806/Full\\_report\\_ENERNUM\\_MAY\\_2019-eng.pdf](https://www.institutparisregion.fr/fileadmin/NewEtudes/Etude_1806/Full_report_ENERNUM_MAY_2019-eng.pdf)
- [3.3] <https://www.ecoinfo.cnrs.fr>
- [3.4] [https://theshiftproject.org/wp-content/uploads/2020/10/Deployer-la-sobriete-numerique\\_Rapport-complet\\_ShiftProject.pdf](https://theshiftproject.org/wp-content/uploads/2020/10/Deployer-la-sobriete-numerique_Rapport-complet_ShiftProject.pdf)
- [3.5] <https://www.greenit.fr/empreinte-environnementale-du-numerique-mondial/>
- [3.6] <https://hal.archives-ouvertes.fr/hal-02549565>
- [3.7] <https://eco-calculateur.dta.aviation-civile.gouv.fr/>
- [3.8] <https://ecolab.ademe.fr/apps/transport>
  
- [5.1] <https://ll-dd.ch>
- [5.2] <https://dmp.opidor.fr>
- [5.3] [http://www.france-grilles.fr/wp-content/uploads/2018/04/ModeleSMP\\_PRESOFTV3.2.pdf](http://www.france-grilles.fr/wp-content/uploads/2018/04/ModeleSMP_PRESOFTV3.2.pdf)
- [5.4] <https://hal.archives-ouvertes.fr/hal-01241196>
  
- [6.1] <https://arxiv.org/pdf/2009.11295.pdf>
- [6.2] <https://greenlab.di.uminho.pt/wp-content/uploads/2017/10/sleFinal.pdf>
- [6.3] <https://github.com/greensoftwarelab/Energy-Languages>
- [6.4] <https://rdcu.be/ciOOJ>
- [6.5] <https://github.com/paugier/nbabel>
  
- [7.1] <https://www.force11.org/group/fairgroup/fairprinciples>
- [7.2] <https://op.europa.eu/en/publication-detail/-/publication/7769a148-f1f6-11e8-9982-01aa75ed71a1/language-en>
- [7.3] <https://www.ovvirlascience.fr/portail-web-de-leosc/>
- [7.4] <https://www.rd-alliance.org/fair>
- [7.5] <https://hal.archives-ouvertes.fr/hal-01241196>
- [7.6] <https://dmp.opidor.fr/>
- [7.7] <https://mi-gt-donnees.pages.math.unistra.fr/site/>
  
- [8.1] <https://hal.archives-ouvertes.fr/hal-01192692/document>
  
- [9.1] [http://www.sti.uniurb.it/events/sfm10qap/slides/katinka\\_slides.pdf](http://www.sti.uniurb.it/events/sfm10qap/slides/katinka_slides.pdf)
  
- [11.1] [https://www.barroso.org/publications/ieee\\_computer07.pdf](https://www.barroso.org/publications/ieee_computer07.pdf)
- [11.2] <https://www.softwareheritage.org/save-and-reference-research-software/?lang=fr>
- [11.3] [https://fr.wikipedia.org/wiki/Long-term\\_support](https://fr.wikipedia.org/wiki/Long-term_support)
- [11.4] <https://ecoinfo.cnrs.fr>
- [11.5] <https://www.ademe.fr/sites/default/files/assets/documents/guide-pratique-face-cachee-numerique.pdf>
- [11.6] <https://principles.green>
  
- [12.1] <https://pop-coe.eu>
- [12.2] [https://fr.wikipedia.org/wiki/Loi\\_d'Amdahl](https://fr.wikipedia.org/wiki/Loi_d'Amdahl)
- [12.3] [https://fr.wikipedia.org/wiki/Message\\_Passing\\_Interface](https://fr.wikipedia.org/wiki/Message_Passing_Interface)
- [12.4] <https://fr.wikipedia.org/wiki/OpenMP>
- [12.5] <https://en.wikipedia.org/wiki/OpenACC>
- [12.6] <https://en.wikipedia.org/wiki/OpenCL>
- [12.7] <https://www.edari.fr>
  
- [13.1] [https://collectif.greenit.fr/ecoconception-web/115-bonnes-pratiques-eco-conception\\_web.html](https://collectif.greenit.fr/ecoconception-web/115-bonnes-pratiques-eco-conception_web.html)
- [13.2] <http://www.ecoindex.fr/>
- [13.3] <http://www.ecometer.org>

Figure 8: Plaquette Bibliographie