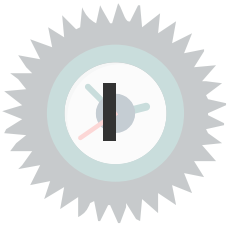
Développements logiciels et ingénierie soutenable

Table des matières

Contexte



1. Contexte

Sources

- *Green Pattern – Manuel d'éco-conception des logiciels – Version 1*, Olivier Philippot, Frédéric Bordage, Thierry Leboucq et al., une action Green Code Lab (greencodelab.fr), 2009.
- *L'écoconception des services numériques*, Caroline Vateau, Sofiann Yousfi Monod, FrédéricBordage et al., GreenIT, 2017.
- *Le numérique en Europe : une approche des impacts environnementaux par l'analyse du cycle de vie*, Frédéric Bordage, Lorraine de Montenay et al., E, GreenIT.fr, 2021.
- *Impacts environnementaux du numérique en France*, Frédéric Bordage, Lorraine de Montenay, Olivier Vergeynst et al., GreenIT, 2021.
- *Rapport annuel du registre des déchets d'équipements électriques et électroniques*, Erwann Fangeat, Alice Deprouw, Marion Jover et al., ADEME, 2020.

Objectifs du développement durable (ONU)

- Définition : un développement qui répond aux besoins des générations présentes sans compromettre la capacité des générations futures à répondre aux leurs.
- A concilier suivant trois éléments de base : croissance économique, inclusion sociale et protection de l'environnement.
- 17 objectifs classés dans 5 domaines :
    - humanité : 1. pas de pauvreté, 2. faim zéro, 3. bonne santé et bien être, 4. éducation de qualité, 5. égalité entre les sexes
    - prospérité : 6. eau propre et assainissement, 7. énergie propre et abordable, 8. travail décent et croissance économique, 9. industrie, innovation et infrastructure, 10. inégalités réduites, 11. villes et communautés durables, 12. consommation et production responsables
    - planète : 13. lutte contre les changements climatiques, 14. vie aquatique, 15. vie terrestre
    - paix : 16. paix, justice et institutions efficaces
    - partenariat : 17 partenariats pour la réalisation des objectifs

Objectifs du développement durable (TIC)

- Pour les TIC (Technologies de l'Information et de la Communication) :
    - réduire **pour** les TIC ou **par** les TIC l'empreinte écologique, économique et sociale
    - inventer un nouveau modèle de société : s'appuyant sur le scénario de l'ONU, prenant en compte les limites de l'écosystème, et en utilisant intelligemment les TIC existantes.

- Trois périmètres :
  - *Green IT 1.0 : réduire l'empreinte des TIC* (API 0051)
  - Green IT 1.5 : réduire l'empreinte des organisations à l'aide des TIC
  - Grren IT 2.0 : inventer de nouveaux produits ou services plus durables grâce aux TIC
- Trois grand smoyens :
  - efficience (techniques plus performantes)
  - sobriété (ressources utilisées avec parcimonie)
  - utilisation des énergies renouvelables

Impacts du numérique sur la planète

Facteurs d'impact à considérer (venant des méthodes ACV orientées problèmes) :

- *atteintes aux ressources abiotiques* : comprenant l'utilisation des minéraux et terre rares, l'utilisation des ressources fossiles, l'utilisation de l'eau potable, etc.
- atteintes aux ressources biotiques : comprenant l'impact direct sur les êtres vivants
- *acidification des océans et des sols*,
- *écotoxicité* : présence de polluants dans les écosystèmes
- *toxicité humaine* : effets néfastes sur la santé humaine
- *eutrophisation* : augmentation des taux d'azote et de phosphore dans les océans et les sols
- *changement climatique*
- *dispersion de radioisotopes* (radiations)
- *destruction de l'ozone stratosphérique*
- *émission d'ozone photochimique et de particules* : apparition de smog
- utilisation des terres

Impacts du numérique sur la planète (ACV du numérique en Europe 2021)

	Niveau 1 : Terminaux utilisateurs	**Niveau 2 : Réseau**	**Niveau 3 : Centres de données**	**Total**
Utilisation des minéraux et terres rares	20.4%	1.4%	1.2%	23%
Utilisation des matières fossiles	10.5%	2.4%	4.1%	17%
Acidification	2.9%	0.5%	1%	4.4%
Ecotoxicité (eau douce)	3.2%	0.5%	1%	4.8%
Toxicité humaine	0.5%	0%	0.1%	0.6%
Eutrophisation	1.4%	0.3%	0.5%	2.2%
Changement climatique	10.6%	1.9%	3.6%	16.1%

	Niveau 1 : Terminaux utilisateurs	Niveau 2 : Réseau	Niveau 3 : Centres de données	Total
Radiations	7.2%	1.6%	2.2%	11%
Destruction de l'ozone stratosphérique	0.1%	0%	0%	0.1%
Emissions d'ozone photochimique et de particules	3.8%	0.7%	1.3%	5.8%

Attention : les terminaux recouvrent tous les outils du numérique. Les plus grands participants sont les téléviseurs, les ordinateurs portables et les smartphones (environ 20% chacun) suivi des ordinateurs de bureau (environ 10%).

Impact du numérique sur la planète (ACV du numérique en France)

		Niveau 1 : Terminaux utilisateurs	Niveau 2 : Réseau	Niveau 3 : Centres de données	Total
Energie	Fabrication	37%	2%	2%	41%
	Usage	27%	19%	13%	59%
GES	Fabrication	76%	5%	2%	83%
	Usage	8%	5%	4%	17%
Eau	Fabrication	86%	1%	1%	88%
	Usage	1%	4%	3%	12%
Matières premières	Fabrication	79%	15%	6%	100%
	Usage	0%	0%	0%	0%

Attention :

- Les matières premières ne prennent pas en compte l'utilisation de ressources fossiles consommées en énergie.
- Sont omises les phases de transport et de fin de vie (pouvant être trouvées dans l'ACV précédent et globalement minoritaires).

Le matériel cause directe

- Empreinte environnementale des TIC principalement causée par la fabrication, et l'utilisation.
- Nécessité d'utiliser le matériel le plus longtemps possible.
- Observation en pratique d'une obsolescence accélérée : durée d'utilisation d'un ordinateur divisée par 4 en 25 ans pour atteindre moins de 3 ans en 2005.

    - Améliorations récentes : dans le milieu professionel, ce chiffre est repassé à 7 ans en 2021 grâce aux lois sur le reconditionnement. Mais cela ne veut pas dire que les entreprises gardent plus longtemps leurs ordinateurs : le marché du reconditionnement cible les particuliers (-30 à -50% de prix d'avhat, 6 mois à 1 an de garantie).
- Causes de l'obsolescence : exigences toujours plus élevées des logiciels, évolution des standards, obsolescence programmée.
- Pour la fin de vie du matériel : rapport sur le traitement des déchets d'équipements électriques et électroniques (DEEE) de l'ADEME
    - DEEE ménagers : insuffisance du recyclage de ces équipements, malgré une nette progression : en 2012 : 5kg traités sur 24 kg d'équipements électroniques jetés par an par français ; en 2019 : 11,6 kg traités sur 22kg d'équipements jetés par an par français (779785 tonnes totales).
    - DEEE professionels : 77% traités en 2019 (sur 75121t), mais attention, tout n'est pas valorisable dans le recyclage.

Le logiciel cause indirecte

- Principal responsable de l'obsolescence accélérée du matériel informatique : exigences toujours croissantes des logiciels.
- Nouvelle version tous les 2 à 3 ans maximum ; fin du support technique d'une version de 3 à 5 ans en moyenne.
- Chaque nouvelle version nécessite en moyenne 2 fois plus de ressources que la précédente.
    - 70 fois plus de RAM pour écrire le même texte entre Windows 7 +Office 2010 par rapport à Windows 98 +Office 97 pour 12 ans d'écart.
    - Phénomène **obésiciel** ou *bloatware*.

Obésiciel

- En pratique, perte de performances entre des versions différentes de logiciel. Deux causes possibles : parasitage ou entropique.
- Obésiciel de parasitage : consommation de ressources par des logiciels non-nécessaires.

  Exemples : logiciels de constructeurs de PC, logiciels en version d'évaluation, utilitaires redondants.
- Obésiciel entropique : perte de performance du logiciel/système d'exploitation considéré aprés changement de version.

  Parmi les raisons, le foisonnement de fonctionnalités inutiles : en moyenne, seulement 20% des fonctionalités d'une application sont utilisées par 80% des utilisateurs.

Aspects économiques (2012)

- Résultat d'une analyse commandée par HP :
    - 5,8% des budgets informatiques consacrés à des applications sous-utilisées.
    - 15% des applications métiers sous-utilisées voire sans apport réel.
    - coût total par an de 16 milliars de dollars en Europe.
- Résultats d'une analyse Opinion Matters :
    - 10% des logiciels achetés ne sont jamais utilisés.
    - 70% des entreprises achètent plus de licences que nécessaires, 80% des décideurs admettent que des logiciels sur les postes de travail ne sont jamais utilisés.
    - coût de 27 milliards de dollars par an aux Etats-Unis et en Grande Bretagne.

    - causés principalement par une mauvaise gestion des actifs logiciels : 60% des entreprises utilisent un tableur, 10% sur papier, 12% ne gèrent rien.
- Le cloud : solution ou problème ?
    - permet de mutualiser les moyens informatiques dans des data-centers proposant un faible PUE (*Power Usage Effectiveness*)
    - effets rebonds :
        - demande d'intégrité forte pour le service : implique des duplications ou triplications matérielles.
        - pour économiser l'énergie, les data centers remplacent leurs serveurs bien avant leur fin de vie.

Durabilité vs Greenwashing

- Le greenwashing consiste à donner une image écologique à une marque ou société en mentant, cachant la vérité ou faisant des allégations non-vérifiées sur des bénéfices environnementaux.
- La société TerraChoice (gestionnaire de l'écolabel canadien Ecologo) définit 7 péchés de greenwashing :
    - compromis caché : prétention ne considérant qu'un nombre restreint d'attributs et en occultant le reste.

      Exemple : appareils électriques économes en énergie, occultant la partie fabrication et fin de vie.
    - absence de preuve : prétention non étayée par une information facile à trouver et digne de confiance.
    - imprécision : prétention vague ou floue.

      Exemple : produit dit *vert* ou *préservant l'environnement* sans définitions de ces termes.
    - non pertinence : prétention exacte mais inutile ou insignifiante.

      Exemple : produit mettant en avant la conformité RoHS (*Restriction of Hazardous Substance*) qui est obligatoire en Europe pour être commercialisé.
    - moindre mal : prétention exacte, mais sur une catégorie de produits globalement nocifs.

      Exemple : la cigarette au tabac biologique qui est *moins mauvaise* pour la santé.
    - faux ecolabel : utilisation de labels internes à l'entreprise peu contraignants et non délivrés par un organisme tiers.

      Exemple nombreux dans les TIC : label GreenIT de Fujitsu Siemens, labels EcoGreenIT et ECOSustainability de NEC, etc.
    - mensonge : prétention fausse.
- D'aprés une étude TerraChoice, moins de 1% des produits évalués étaient concernés par le péché de mensonge, mais 73% étaient concernés par le péché de compromis caché.

Conception

1. ACV

Principe et définitions

- Objectif : identifier l'impact environnemental d'un produit **tout au long de son cycle de vie**.
- Définition : l'ACV (Analyse Cycle de Vie) consiste à inventorier les flux de matières et d'énergies entrants et sortants à chaque étape du cycle de vie d'un produit pour ensuite évaluer ses impacts environnementaux.
- Normalisé en 1997 dans les normes ISO 14040.
- Principales phases de l'ACV : extraction des matières premières et fabrication, transport et distribution, usage, fin de vie
- critères environnementaux : changement climatique, destruction de l'ozone stratosphérique, acidification et eutrophisation des océans, formation d'agents photo-oxydants (smog), atteinte et épuisement des ressources biotiques et abiotiques, utilisation des terres, impacts sur la santé humaine.
- ne pas oublier les aspects économiques et sociaux qui ne sont pas traiter par l'ACV : respect des principes équitables dans sa phase de production et commercialisation, discrimination ou manque d'accessibilité, viabilité économique, impacts sociaux.

ACV du logiciel

- Difficile à réaliser : pas de déchets visibles générés, pas d'obsolescence intrinsèque, provenant de sources de production liées au matériel et aux ressources humaines mobilisées lors du développement.
- Phases d'un ACV logiciel : fabrication, distribution, utilisation, fin de vie.
- Principales problématiques :
    - ressources de développement généralement partagées sur plusieurs projets : nécessité de bien suivre le travail effectué sur chaque logiciel.
    - fabrication non terminée à la commercialisation : la maintenance comprend en moyenne 50% des coûts de développement.
    - une nouvelle version est-elle une continuation ou un nouveau produit ?
    - des parties du logiciel peuvent être des bibliothèques existantes ou développées par des sous-traitants.

Phase de fabrication

- Principaux impacts environnementaux :
    - déplacements,
    - fonctionnement d'une entreprise de service dématérialisée (moyens humains et matériels pour le développement).

- Actions recommandées (typique pour les entreprises de service dématérialisées) :
    - favoriser le télétravail pour réduire les déplacements,
    - prolonger la durée de vie du matériel et des infrastructures,
    - favoriser la gestion de configuration,
    - économiser les consommables et l'énergie sur le matériel,
    - mettre en place un cadre de travail agréable ,
    - favoriser la non-discrimination.

Phase de distribution

- Principaux impacts environnementaux :
    - moyens techniques (serveurs, réseaux, terminaux, etc.)
    - moyens humains
    - éventuellement, affrètement et production de manuels utilisateurs et supports matériels (DVD).
- Actions recommandées (typique pour les entreprises de bien de consommation) :
    - privilégier une distribution dématérialisée,
    - éviter des installations de trop grosses tailles en espace disques et des performances surdimensionnées,
    - permettre une installation fonctionnelle modulaire pour diminuer l'espace disque et mémoire requis,
    - respect des données personnelles,
    - pratiques commerciales claires pour le client.

Phase d'utilisation

- Principaux impacts environnementaux :
    - énergie consommée lors de l'utilisation,
    - production des équipements nécessaire sà l'utilisation du logiciel (serveurs, terminaux, etc.)
    - consommables,
- Actions recommandées :
    - bonne conception pour éviter l'obésiciel et une consommation importante de ressource :
        - étudier les choix d'architecture du logiciel et décomposer les fonctionnalités par service,
        - bien choisir les langages et bibliothèques, favoriser la réutilisation, et optimiser (quitte à re-développer) les fonctions majeures et consommatrices,
        - mettre en place des outils de mesure de la qualité et de la consommation pour favoriser la maintenance,
        - optimiser la volumétrie des données nécessaires.
    - considérer les aspects sociaux :
        - accessibilité au plus grand nombre,
        - privilégier l'usage : le logiciel doit d'adapter à l'usage et ne proposer que les fonctionnalités nécessaires,
        - sûreté et sécurité du produit,

        - respect des données personnelles.

Phase de fin de vie

- Principaux impacts environnementaux :
    - désinstallation,
    - fin de vie des données (recyclage, réutilisation, anonymisation) ou de l'accés à un service (désinscription)
- Actions recommandées :
    - pérennisation du code (documentation et évolutivité),
    - désinstallation propre,
    - récupération facile des données,
    - interruption du logiciel ou du service sans contrepartie dissuasive.

2. Ethique et normes

Cycles de vie et durabilité

- Généralement, éviter la programmation *quick and dirty*.

  Une dette technique se transforme généralement en dette environnementale : obésiciel, code non maintenu et abandonné, code mal optimisé, etc.
- L'aspect environnemental peut être facilement intégré comme exigences dans la plupart des modèles de cycle de vie.

  Exemple du cycle en V : lister les exigences environnementales dans la phase descendante de l'expression des besoins, les dériver tout au long de la phase descendante et vérifier leur intégration dans la phase ascendante.
- Avantages environnementaux dans les méthodes agiles :
    - meilleure convergence avec les besoins utilisateurs (pas de fonctions inutiles, retour rapide du client sur le produit, meilleure maîtrise des coûts et des délais en pouvant arrêter le projet à tout moment),
    - processus itératif permettant d'ajouter si nécessaire à chaque cycle au backlog des tâches de réduction de la dette technique et de mesures et corrections des contraintes environnementales.

Accessibilité

- Des normes d'accessibilité existent, mais sont souvent peu appliqués ou seulement aux sites internet.
    - normes d'accessibilité de sites itnernet développés par le World Wide Web Consortium (W3C),
    - normes pour documents électroniques développés par le Consortium Daisy
    - norme ISO 9241 pour l'amélioration de l'accessibilité de l'équipement, des services des TIC et des logiciels.

      Elle préconise en particulier :
        - une information perceptive par les utilisateurs,
        - des contenus compréhensibles par les utilisateurs,
        - des interfaces utilisables facilement,
        - des logiciels tolérants aux fautes,

        - une flexibilité permettant des choix d'entrées et sorties alternatifs.
- Outils et aides pour les développeurs :
    - checklists pour vérifier la couverture d'items d'accessibilités (Voluntary Product Accessibility Template de l'information Technology Industry Council, Checklist d'accessibilité IBM),
    - API pour des fonctions d'accessibilités (Java Accessibility Utilities, IAcessible2 pour Linux),
    - outils d'émulation (outil FANG pour émuler les lecteurs d'écran)
- méthodes d'évaluation de l'acessibilité d'un logiciel :
    - tests automatiques de règles
    - évaluation par des experts
    - évaluation par émulation
    - évaluation par des utilisateurs réels

Code d'éthique

- Appétence de plus en plus importante des ingénieurs pour un impact positif et durable de leur travail sur la société.
- Proposition conjointe de code d'éthique des développeurs par l'ACM (Association for computing) et l'IEEE-CS.

  Principes fondamentaux en faveur de :
    - Public : Les ingénieurs logiciels doivent agir en conformité avec l'intérêt public.
    - Client et l'employeur : Les ingénieurs logiciels doivent agir d'une manière qui est dans le meilleur intérêt de leur client et l'employeur et conforme à l'intérêt public.
    - Produit : Les ingénieurs logiciels doivent s'assurer que leurs produits et modifications connexes sont conformes aux plus hautes normes professionnelles possibles.
    - Jugements : Les ingénieurs logiciels doivent maintenir l'intégrité et l'indépendance de leur jugement professionnel.
    - Gestion : Les responsables de l'ingénierie du logiciel et les dirigeants doivent souscrire et promouvoir une approche éthique de la gestion du développement de logiciels et de la maintenance.
    - Profession : Les ingénieurs logiciels doivent porter la réputation de la profession conformément à l'intérêt public.
    - Collègues : Les ingénieurs logiciels doivent être justes et aider leurs collègues.
    - Soi-même : Les ingénieurs logiciels doivent participer à l'apprentissage tout au long de la pratique de leur profession et promouvoir une approche éthique de la pratique de la profession.

Développement

1. Principe des Green Design Patterns

Principe

- Objectif : s'inspirer des design patterns de la conception logicielle pour proposer des principes généraux de conception écologique.
- Définition : un green design pattern est un motif de conception logiciel réutilisable qui a pour but de réduire l'impact environnemental du système sur lequel il sera installé.
- Attention à ne pas laisser de côté les aspects sociaux et économiques.

Leviers d'amélioration

- 6 grands types d'amélioration :
  - maîtrise de l'impact de la phase de fabrication,
  - réduction de la consommation d'énergie,
  - réduction des ressources nécessaires,
  - augmentation des performances,
  - facilité d'utilisation,
  - prolongation de la durée de vie du logiciel
- leviers d'amélioration :
  - efficacité d'architecture : langage, calculs distribués, etc.

    Exemple : diminution de 50% des traitements par FaceBook en passant du PHP à du C++ compilé automatiquement

    → réduction de la consommation d'énergie, réduction des ressources nécessaires, augmentation des performances
  - efficacité de calculs : algorithmes plus performants

    → réduction de la consommation d'énergie, augmentation des performances
  - efficacité de données : utilisation de caches plutôt que de lire continuellement en RAM ou ROM, défragmentation des fichiers, etc.

    → réduction de la consommation d'énergie, réduction des ressources nécessaires, augmentation des performances
  - Efficacité de prise en compte de contexte : proposer des sorties ou des traitements adaptées au contexte d'utilisation

    → facilité d'utilisation, réduction de la consommation d'énergie, réduction des ressources nécessaires
  - efficacité des outils : outils de qualimétrie, méthodes de projet adaptées

    → maîtrise de l'impact de la phase de fabrication

- pour prendre les bonnes décisions, besoins de connaissances sur les composants logiciels et matériels des ordinateurs.

2. Exemples de Green Pattern

Efficacité d'architecture

- Efficacité d'unité : découpage du logiciel en unités aussi indépendantes que possibles pour réaliser des traitements cohérents en espace et en temps et simplifier la compréhension et la programmation.
    - Exemple : utilisation de design pattern pertinents (MVC, Multi-tier, Pair-à-pair, etc.). Attention à ne pas utiliser des design pattern non adaptés !
- Efficacité de répartition : distribution adéquate des unités sur les éléments physiques.
    - Exemple : distribution cohérente des traitements en terme de calcul/communication pour des applications client/serveur.
- Abstraction des couches basses : regroupement des traitements de même fréquence dans une même couche pour les faciliter et simplifier la programmation ; échange de données uniquement avec les couches connexes.
    - Exemple : séparation des traitements sur des PC : couche basse (driver), couche intermédiaire (OS), couche applicative (applications logicielles). Peut être contradictoire avec l'efficacité de calcul !
- Modularité : modularité du logicielle pour n'offrir que les fonctionnalités nécessaires.
    - Exemple : modules tiers (Firefox, Eclipse, etc.), modularité de l'installation du logiciel.

Efficacité de calcul

- Multi-tâche : utilisation des mécanismes de threading des OS généralement optimisés et robustes.
    - Bonnes pratiques : ne pas abuser des blocs d'instruction atomiques ; privilégier des méthodes courtes ; mutualiser les traitements pour libérer rapidement le processeur le la mémoire ; mutualiser les entrées/sorties pour libérer rapidement les drivers.
- Multi-coeur : utilisation des calculateurs multi-coeur pour optimiser le traitement. Attention : processus complexe donc risque d'un effet inverse.

  Deux aspects à considérer :
    - décomposition des traitements : division des données ou division des fonctionnalités,
    - répartition des traitements : répartition équilibrée ou déséquilibrée
- Tâches périodiques : limiter l'utilisation de timers et de tâches périodiques, consommatrices en ressource et empêchant le passage en mode veille. Préférer les tâches plus longues mais de périodicité plus faible.
- Boucles : utilisation à limiter à cause de leur consommation en ressources.
    - Bonnes pratiques : optimiser la condition de sortie de la boucle, ne pas utiliser pour surveiller des données.
- Bibliothèques : utiliser des bibliothèques existantes (de préférence robuste et optimisée pour le matériel) pour simplifier le développement et optimiser des fonctionnalités.

  Attention, la généricité des bibliothèques peut impacter négativement l'utilisation de ressources.
    - bonnes pratiques : faire des POC et des benchmarks pour choisir la meilleure bibliothèque.

- Compilateur : bien choisir le compilateur et ses options.
    - exemple : suivant les options matérielles choisies sur gcc, jusqu'à 4 fois plus de performance.
- Driver : importance cruciale des performances des drivers.

Efficacité de données : patterns techniques

- Distance aux données : bien utiliser la gestion de cache et mémoire pour accéder rapidement aux données nécessaires.

  Exemple de distances :
    - registre CPU : 1 cycle, 10 mots par cycle
    - cache L1 (CPU) : 3 cycles, 2 mots par cyclr
    - RAM : 100 cycles, 0.5 mot par cycle
    - HDD : 1 million de cycles , 0.1 mot par cycle
- Cache : optimisation pour limiter les pertes de cache.

  3 grandes techniques :
    - réutilisation de données : regrouper les traitements de données pour éviter des rechargements intempestifs
    - réduire les formats de données
    - réarranger les données : réorganisation des structures, fusion des tableaux, séparation hot/cold splitting
- Coût des données : limiter les bits de bourrage (padding) ajoutés par les compilateurs.
    - en C sur un compilateur 32 octets, la structure {bool b, double d, short d, int i} prend 22 octets alors que {double d, int i, short s, bool b} n'en prend que 16.
    - sur une JVM 32 octets, le surcoût d'un double est de 16 octets (soit 200% des données utiles), le surcoût d'un TreeMap <double, double> de 100 entrées est de 82%.
- Optimisation des I/O : optimisation dépendante des caractéristiques techniques du périphérique utilisé.
    - pour un HDD : augmenter la taille des blocs pour diminuer la consommation d'énergie, préférer les opérations synchrones, travailler sur un disque dur défragmenté.

Efficacité de données : patterns fonctionnels

- Réduction des accés disques.
- Compression des données : attention à ce que l'économie de place ne soit pas contrebalancée par des coûts de traitement.
- Regroupement des traitements de données : permet l'optimisation du cache ou des accés disques.
- Gestion de la durée de vie des données : bien penser à la destruction des données pendant l'opération et en fin de vie.
- Utilisation de technologies adaptées à la fonctionnalité.

  Exemple : utiliser des bases de données stoquées en mémoire plutôt qu'en disque pour optimiser l'accés aux données, utiliser des pages internet simples plutôt que basées sur des systèmes de gesion de contenu pour limiter l'utilisation mémoire.

Efficacité de prise en compte du contexte

- Transitions de mise en veille : une des prises en compte de contexte les plus importantes pour économiser l'énergie.
- Gestion de batterie : prendre en compte les états de batterie en charge/décharge, état de la batterie, moniteur allumé/éteint.
- Etat du système : prendre en compte les aspects environnementaux du logiciel : état de la connexion réseau, luminosité ambiante, état micro/audio, etc.

Phase opérationnelle et fin de vie

1. De la validation à la fin de vie

Validation

- Actions qualité
  - Maîtrise de l'impact du processus de développement : avoir de bonnes procédures de tests pour détecter au plus tôt les fautes et les corriger de la façon la moins coûteuse.
  - Vérification de l'implémentation des green patterns : Mise en place d'un plan de validation pour vérifier les objectifs environnementaux spécifiés.
  - Amélioration de la qualité logicielle : impact direct sur de nombreux aspects environnementaux (réutilisabilité, satisfaction client, etc.).
- Mesures de la consommation énergétique

  Mesurée en Wh (Wattheure) : puissance*temps = (pour le courant continu) tension*intensité*temps
  - appareillage physique : wattmètres
  - outils logiciels directs : Powertutor (visant les applications pour téléphone Android), Green Fox (module Firefox visant les pages ou applications web), Intel Energy Checkser (SDK permettant de se connecter à des instruments de mesure).
  - outils logiciels indirects : Ptop (outil GNU/Linux pour évaluer la consommation à partir d'un modèle), Microsoft Event Tracing for Windows (ETW), Powertop (analyse du temps passé par les applications GNU/Linux dans les différents états du processeurs).
- Evaluation de l'impact global : penser à réévaluer l'impact environnemental globale tout au long du projet mais aussi sdu cycle de vie de l'application

  Exemples :
  - ne pas oublier les POC ou les bibliothèques utilisées.
  - une modification dans le projet (changement du langage, de l'architecture, etc.) nécessite une rééavaluation de l'impact.
  - un changement pendant le cycle de vie (nouvelle version, augmentation du nombre de déploiements) nécessite une réévaluation de l'impact.

Diffusion

- Canaux de diffusion
  - fabrication / diffusion des supports matériels : utilisation de matières premières, afffrêtement, recyclage, etc.
  - espace de stoquage pour diffusion électronique
  - charge réseau pour diffusion électronique
  - manuel papier / électronique

Attention à la vérification automatique de mise à jour (consommatrice en bande passante et performance) !

- Diminution des éléments annexes
    - Diminuer la taille des boîtes des supports physiques.
    - Ne pas inclure différents types de plastique pour faciliter le recyclage.
    - si possible préférer une diffusion numérique de la documentation., si possible en local (CD, support USB) plutôt que sur un serveur.

Fin de vie

- Valorisation des éléments gérés par le logiciel :

  Prendre en compte :
    - l'exécutable,
    - les fichiers nécessaires à l'exécutable (driver, bases de données, etc.),
    - les fichiers installés dans le système d'exploitation (registre, etc.),
    - les éléments en mémoire,
    - les fichiers générés (fichiers temporaires, profils, raccourcis, etc.).
- Valorisation des éléments gérés par l'utilisateur (sorties du logiciel) :
    - proposer un archivage avec compression de données (par exemple : mails)
    - proposer un décommissionement aprés une date d'expiration (exemple : machines virtuelles)
    - proposer des sorties interopérables et donc réutilisables
- Mettre en place des outils de détection et désinstallation de logiciels non utilisés.