

Inclusion des enjeux de maîtrise du système d'information dans les politiques numériques responsables

François Pellegrini, francois@pellegrini.cc

Version v1.0a, 17/12/2025

1 Contexte

Le « numérique responsable » vise à réduire l'empreinte sociale, environnementale et économique du numérique, en agissant tant au niveau des technologies que des usages. Puisque 75 à 80 % de l'empreinte environnementale du numérique provient de l'étape de fabrication des matériels, l'augmentation de la durée de vie et de la réutilisabilité des matériels est un facteur dominant de réduction de l'empreinte environnementale du numérique. Or, un matériel numérique actif (c'est-à-dire disposant d'une logique de contrôle programmée) ne peut être exploité sans le logiciel lié qui le pilote.

1.1 Renouvellement imposé

Si le progrès technologique peut justifier le retrait anticipé d'un parc fonctionnel, en cas d'amélioration significative des performances de la génération suivante (on peut penser au remplacement des écrans cathodiques par les écrans LCD), une décision de renouvellement « responsable » doit rester à la main des personnes concernées, qui ont à en supporter le coût et les externalités. Cette décision ne peut être dictée par les fournisseurs, sur la base d'intérêts parfois contraires à ceux des organisations touchées. Or, par le passé, de tels diktats ont eu lieu, comme lors de la sortie de Windows Vista, dont la voracité en ressources (puissance de calcul et mémoire) a imposé à de trop nombreuses organisations de remplacer leur matériel informatique, même récent, par un matériel suffisamment puissant pour exécuter ce nouveau logiciel.

1.2 Sources d'obsolescence

Les entraves à la poursuite du fonctionnement des systèmes numériques sont nombreuses.

La première est l'obsolescence matérielle. Elle peut prendre la forme du choix de composants à faible durée de vie lors de la fabrication d'un matériel, ou bien du choix d'architectures et de procédés de fabrication nuisant à la réparabilité (par exemple, soudure plastique au lieu de vissage). Il s'agit d'une question maintenant bien identifiée au niveau des acteurs du numérique responsable et des pouvoirs publics, qui cherchent à peser sur le marché en produisant des indicateurs de réparabilité et en imposant des valeurs minimales pour les achats à venir, par exemple.

La seconde est l'obsolescence logicielle. Cette dernière est actuellement trop peu prise en compte dans les politiques « numérique responsable », voire pas du tout. Or, ses effets sur la mise au rebut des matériels sont considérables.

1.2.1 Obsolescence logicielle programmée

L'obsolescence logicielle est majoritairement une « obsolescence programmée » : si, dans quelques cas, l'introduction d'algorithmes plus complexes ou de modèles mathématiques plus gourmands en données nécessite d'acquérir immédiatement un matériel plus puissant, dans la majorité des cas, c'est la fin du support d'un logiciel par l'éditeur ou l'intégrateur qui signe l'arrêt de mort d'un système. Cette mort peut être lente, matérialisée par une perte progressive de compatibilité avec les nouvelles versions des systèmes environnants en adhérence, ce qui causera une gêne croissante à l'usage ; c'est par exemple le cas lorsque l'absence de support des nouvelles versions de formats de fichiers oblige les utilisateurs à effectuer des conversions vers les anciennes versions pour que le système fonctionne. Cette mort peut aussi être brutale, lorsque la montée en version des logiciels environnants provoque une fin brutale de compatibilité. C'est le cas également lorsqu'une vulnérabilité est découverte dans une version de logiciel qui n'est plus maintenue : les opérateurs du système, face à une telle occurrence, imprévisible et brutale, doivent alors immédiatement déconnecter le système en question. Afin de ne pas se retrouver dans cette situation et pour réduire la surface d'attaque, les systèmes sont donc majoritairement retirés du service et remplacés dès que la maintenance de leurs logiciels n'est plus assurée. L'existence d'une offre commerciale de support de longue durée (LTS, pour « *long-term support* ») ne suffit cependant pas à résoudre le problème : ces offres sont en général elles aussi limitées dans le temps, et renchérissent le coût des systèmes par rapport à leur remplacement.

1.2.2 Obsolescence logicielle déclenchée

L'obsolescence programmée peut également être une « obsolescence déclenchée ». On désigne ainsi le fait que l'obsolescence advienne brutalement, par la décision d'un tiers, comme lors de l'arrêt de serveurs alimentant des terminaux : ceux-ci sont rendus irrémédiablement inutilisables. C'est notamment le cas des services attachés aux consoles de jeu ou, récemment, du déclenchement à distance par Apple d'un mécanisme rendant inopérants tous les appareils vendus en Russie et dont les composants de remplacement ont été considérés par cette société comme « non authentiques ». Cet exemple illustre l'usage de techniques de marquage des composants par des éléments codés permettant d'en tracer individuellement la provenance. Cette pratique permet également d'entraver la commercialisation de composants compatibles, la copie conforme de ces éléments, nécessaire à leur acceptation par le système, étant poursuivie en tant que violation du droit d'auteur de l'entreprise sur ces éléments de forme (ce détournement du droit d'auteur est appelé « *para-copyright* » dans la littérature).

L'obsolescence déclenchée constitue un risque écologique et stratégique majeur pour les organisations, car elle n'est pas prévisible et constitue un risque grave d'atteinte à la maîtrise des systèmes d'information, portant de ce fait des enjeux de souveraineté numérique.

1.3 Lutter contre l'obsolescence programmée

L'obsolescence programmée est évitable. Elle résulte d'un contrôle excessif des éditeurs de logiciels sur les fonctionnalités et la feuille de route de ceux-ci. Ce déséquilibre empêche les clients et usagers de faire valoir leur droit à la « jouissance paisible » de leurs systèmes informatiques, manifestant de façon éclatante la pertinence de l'aphorisme de Lawrence Lessig, « *Code is law* » : ce qui a été programmé par un tiers s'impose aux usagers comme une loi à laquelle on ne peut déroger.

L'obsolescence programmée constitue un phénomène avéré et coûteux, mais non mesuré dans les études d'usage du numérique, notamment dans le secteur des terminaux mobiles, alors qu'ils y sont largement soumis. Le renouvellement encore actuellement très (trop) rapide des terminaux mobiles masque le phénomène, les usagers changeant de matériel bien avant que les éditeurs ne mettent fin au support des logiciels. Seuls les rares usagers souhaitant faire durer leurs équipements au-delà de cinq à sept ans se trouvent confrontés à la fin du support de leurs systèmes d'exploitation, qui conduira à des difficultés progressives d'installation des nouvelles versions des applications mobiles.

Or, ce sujet a déjà été identifié, sans pour autant être traduit en actions. Ainsi, le rapport de l'ADEME sur le numérique mentionne-t-il que : « L'extension de la durée de vie des équipements informatiques, encouragée par l'utilisation de systèmes d'exploitation moins gourmands comme Linux, pourrait réduire les déchets électroniques de 30 % à 50% », en permettant de gagner 2 à 4 années de vie supplémentaires sur les terminaux. Une étude sur les performances respectives des systèmes d'exploitation indique pour sa part que, à tâches égales, un système Linux conduit à économiser 15 à 20 % d'énergie comparé à un système Microsoft Windows. Le recours à Linux est souvent rejeté selon l'argument que ces systèmes seraient minoritaires en nombre et donc ne seraient pas en mesure d'offrir l'éventail de solutions applicatives que l'on trouve sous Microsoft Windows, par exemple. Ceci est partiellement vrai, mais la situation est en évolution rapide. Outre l'existence d'émulateurs permettant d'exécuter sous Linux des applications conçues pour Microsoft Windows, un signal fort vient de la percée fulgurante de Linux en Inde, où il a représenté, mi-2024, 15 % du marché des stations de travail. L'émergence d'un tel parc, dans un pays de 1,7 milliards d'habitants, est de nature à pousser les éditeurs à offrir leurs logiciels nativement sur cette plateforme, à court terme.

2 Problématique

Le coût écologique et économique induit par le manque de maîtrise des systèmes d'information n'est actuellement pas pris en compte dans les outils de pilotage des politiques « numérique responsable » des organisations.

Cette prise en compte, pour intégration au sein des feuilles de route « numérique responsable », nécessite de disposer :

1. d'une méthodologie d'évaluation ;
2. d'indicateurs adaptés, conformes aux préceptes « SMART », c'est-à-dire : Spécifiques, Mesurables, Acceptables, Réalistes, et Temporellement définis ; et

3. de moyens d'action au sein de l'organisation.

Le présent document a pour objectif de proposer une méthodologie concernant les points 1 et 2 ci-dessus, permettant de contribuer à l'inclusion des critères de maîtrise des systèmes d'information au sein des outils de pilotage d'une feuille de route numérique responsable. Il propose un cadre conceptuel, une méthodologie d'analyse, et la définition d'indicateurs individuels, collectifs et global permettant de calculer des scores de maîtrise à même d'orienter un plan d'action.

3 Concepts et terminologie

Il y a **dépendance** entre deux composants si le remplacement de l'un par un composant tiers nécessite le remplacement de l'autre pour garantir le fonctionnement du système. C'est le cas de logiciels conçus pour un système d'exploitation donné, en l'absence d'émulateurs. Il s'agit d'un levier de pression majeur sur les usagers, au sein des marchés captifs.

Il y a **adhérence** si le remplacement d'un composant conduit à une dégradation de la qualité des services rendus par l'autre. Tel est le cas d'un agenda partagé qui n'offre un greffon qu'avec certains systèmes de visio-conférence, à l'exclusion d'autres systèmes.

Il y a **compatibilité** quand plusieurs produits peuvent fonctionner ensemble sans que les usagers puissent avoir connaissance de la façon dont les produits interagissent (formats de données et protocoles fermés) ni ne puissent intervenir pour remplacer l'un des produits par celui de leur choix. La compatibilité résulte d'un accord contractuel entre les concepteurs des produits, qui exclut les usagers.

Il y a **interopérabilité** quand plusieurs produits peuvent fonctionner ensemble et que les modes de leur interaction sont connus, grâce à l'existence d'un format ou protocole de communication ouvert. Ces spécifications sont **ouvertes** lorsqu'elles sont publiques et librement réutilisable par chacun (comme par exemple le format PDF), et **libres** lorsque leur définition et leur évolution font l'objet d'un processus de gouvernance ouvert à toutes les parties prenantes intéressées.

Une licence logicielle est considérée comme une **licence libre** si elle garantit simultanément les quatre libertés définies par la Free Software Foundation :

- liberté d'usage du logiciel obtenu ;
- liberté de copie du logiciel obtenu ;
- liberté de modification du logiciel obtenu (ce qui nécessite l'accès au code source) ;
- liberté de redistribuer le logiciel modifié (il ne s'agit pas d'une obligation).

Des licences non libres sont des licences qui ne respectent pas l'une au moins des quatre libertés. Elles seront dites plus ou moins **ouvertes** selon le degré des libertés qu'elles accordent. Elles ne pourront cependant être qualifiées d'« **open source** », ce terme étant relatif à l'*Open Source Definition* en dix critères de l'Open Source Initiative, et correspondant par ses effets à la définition d'une licence libre.

On appelle **logiciel libre** un logiciel dont au moins un canal de distribution accessible aux usagers considérés a recours à une licence libre (sans pour autant que le logiciel soit librement accessible au public sous cette licence, comme par exemple dans le cas d'un consortium fermé).

On appelle **communauté** un vivier de compétences humaines attachées à un bien ou service donné, tel qu'une communauté d'usage, d'assistance, de développement, etc.

On appelle **résilience** la capacité d'un système, tel qu'un logiciel, à survivre à la disparition de son éditeur actuel (entreprise ou communauté). Le simple accès au code source n'est pas une garantie suffisante, car la capacité pour des tiers de reprendre à leur compte le développement du logiciel est conditionnée par les droits concédés par la licence. La résilience d'un logiciel à sources fermés est nulle, sauf dans le cas de dispositions prises par l'éditeur visant à la mise sous séquestre de son patrimoine immatériel, avec transfert aux clients des codes sources et des droits afférents en cas de défaillance.

La résilience d'un logiciel nécessite l'existence de personnes à même d'en reprendre le développement. Un logiciel libre sans communauté est bien moins résilient qu'un logiciel disposant d'une communauté de contributeurs vibrionnante.

4 Méthodologie d'évaluation

Afin de pouvoir être facilement comprise et adoptée, la méthode d'évaluation proposée ne doit pas conduire à collecter et traiter des quantités excessives de données.

4.1 Modalités de mise en œuvre

La collecte doit pouvoir être mutualisée avec une collecte préexistante, dont les données sont utiles à la mise en œuvre de la méthode d'évaluation. En l'occurrence, il s'agit de la cartographie du système d'information, recensant le parc matériel et logiciel utilisé dans l'organisation. L'objectif de maîtrise des systèmes d'information constitue donc une motivation supplémentaire à la construction de cette cartographie. Celle-ci s'enrichit d'informations spécifiques, constituant un graphe d'interdépendances entre matériels, services et logiciels.

Les indicateurs doivent être simples à comprendre. Les résultats fournis doivent être facilement interprétables. Notamment, une présentation par agrégats doit donner une vision synthétique des sujets à traiter en priorité. Les indicateurs doivent être suffisamment sensibles pour qu'une action unique pertinente sur le système d'information conduise à l'amélioration du résultat, rendant ainsi perceptibles les actions entreprises.

Chaque composant du système d'information doit être évalué isolément, sur la base des caractéristiques propres liées à son fournisseur et son écosystème, mais aussi en relation avec les autres composants avec lesquels il interagit, afin d'identifier les dépendances et adhérences.

4.2 Scores de dépendance

Les scores de dépendance calculés sont donc de trois sortes :

- individuels, au niveau de chaque composant matériel et logiciel identifié ;

- collectifs, au niveau de chaque ensemble inter-dépendant identifié grâce au graphe de dépendances issu de la cartographie du système d'information ;
- global, fournissant ainsi un indicateur clé de performance (« *Key Performance Indicator* », ou KPI) aux échelons décisionnels.

Pour des raisons mathématiques (effet de l'opération de multiplication décrit plus bas), les scores calculés sont des scores de risque, et non de conformité : plus les scores sont élevés, et moins le système d'information est maîtrisé ; plus ils sont bas (proches de zéro, en pratique), et plus le système d'information est maîtrisé.

4.3 Propriétés désirables

La méthode de calcul doit exhiber les propriétés suivantes :

- Plus une inter-dépendance mobilise de composants, plus elle est risquée et plus son score doit être élevé. Le score collectif d'une technologie doit donc dépendre de la taille des agrégats formés par les sous-ensembles interconnectés du graphe de dépendances ;
- Certains agrégats sont plus critiques que d'autres. Par exemple, pour une organisation dont le cœur de métier n'est pas la publication, la suite logicielle de production graphique, même constituée d'outils très inter-dépendants, ne doit pas représenter le premier chantier. Un coefficient multiplicateur de criticité doit donc être associé à chaque agrégat, pour en pondérer le score ;
- L'amélioration de la maîtrise d'une fonction non critique ne doit pas systématiquement conduire à la diminution du score global, car cela pourrait donner l'impression que la problématique de maîtrise est en cours de résolution, alors que les points essentiels n'ont pas été traités. Le score global ne peut donc pas être la somme des scores des agrégats, mais plutôt le maximum des scores pondérés des agrégats. Ainsi, le score ne diminuera pas tant que l'agrégat le plus critique n'aura pas été traité, soit en en réduisant la taille (retrait d'un item de l'agrégat, ou réduction de sa criticité par la mise en œuvre de mesures de sécurisation).

4.4 Identification des composants

Soit C l'ensemble des composants du système d'information. Ceux-ci peuvent être soit matériels, soit logiciels.

Dans le cas d'un matériel actif embarquant du logiciel (firmware d'équipement réseau, etc.), l'équipement sera considéré sous la forme de deux composants distincts, vraisemblablement dépendants l'un de l'autre. Cela conduira à ce que deux scores soient calculés : l'un pour la partie matérielle et l'autre pour la partie logicielle.

De même, dans le cas des services de type client-serveur, la partie cliente et la partie serveur seront considérées comme deux composants (ou systèmes multi-composants) distincts. Cela permettra de traiter séparément les questions de dépendance vis-à-vis du serveur et du client.

4.5 Graphe de dépendance

Le graphe de dépendance G du système d'information est construit de la façon suivante. Les sommets de G représentent les composants C_i du système, et il existe dans G une arête entre deux composants C_i et C_j (c'est-à-dire que C_i est relié à C_j , sans direction d'orientation du lien) si C_i dépend de C_j , ou l'inverse.

Les critères de dépendance dépendent du type de composants. Il s'agit le plus souvent d'une dépendance fonctionnelle, telle que, par exemple, la spécificité d'un logiciel vis-à-vis d'un type de système d'exploitation donné, l'existence d'une compatibilité sans interopérabilité entre deux logiciels d'une chaîne (notamment via l'usage de formats de fichiers fermés), etc.

Les adhérences ne constituent pas des dépendances en tant que telles. Cependant, des adhérences fortes, qui pénalisent fortement l'efficacité et la qualité du travail des personnes, peuvent être intégrées au graphe de dépendances afin de représenter un objectif spécifique à traiter.

4.6 Score individuel (matériel)

Le score individuel d'un composant matériel est calculé par sommation des valeurs associées à cinq critères, définis dans la table ci-dessous. Lorsque le critère n'est pas applicable, un score de zéro doit être appliqué au critère correspondant, afin qu'il ne pénalise pas le score.

Critère	Valeur			
	0	1	2	3
Durée de la garantie	Très longue	Longue	Courte	Très courte
Durée de fourniture des pièces détachées (source primaire)	Très longue	Longue	Courte	Très courte
Durée de fourniture des pièces détachées (sources secondaires / réemploi)	Très longue	Longue	Courte	Très courte
Reprogrammable	Oui, librement		Seulement par le logiciel fourni par le fabricant	Non
Connectique*	Normalisée, avec convertisseurs**	Normalisée, sans convertisseurs	Partagée par plusieurs fabricants	Propre au fabricant

* Le score de connectique sera le maximum des scores établis pour chacune des connectiques dont dispose l'appareil, en entrée comme en sortie.

** Les convertisseurs sont des dispositifs annexes permettant de préserver l'interopérabilité avec les technologies plus anciennes, telles que HDMI vis-à-vis de VGA, ou USB-C vis-à-vis de USB-B.

Le score individuel d'un composant matériel est donc compris entre 0 (maîtrise totale du composant) et 15 (composant non maîtrisé).

4.7 Score individuel (logiciel)

Le score individuel d'un composant logiciel est calculé par sommation des valeurs associées à sept critères, définis dans la table ci-dessous. Lorsque le critère n'est pas applicable, un score de zéro doit être appliqué au critère correspondant, afin qu'il ne pénalise pas le score.

Critère	Valeur			
	0	1	2	3
Support	24 / 7	Garanti	Partiel	Aucun
Licence	Libre	Permet la modification locale		Totalement fermée
Code source	Disponible avec outils de mise en production	Disponible sans outils de mise en production	Sous séquestre	Caché
Communauté	Très nombreuse et très active, assistance possible	Nombreuse et active, écoute possible	Existence de plusieurs contributeurs identifiables	Aucune
Format de données d'entrée	Ouvert, gouvernance libre	Ouvert, gouvernance éditeur	Fermé, avec convertisseurs	Fermé
Format de données interne	Ouvert, gouvernance libre	Ouvert, gouvernance éditeur	Fermé, avec convertisseurs	Fermé
Format de donnée de sortie	Ouvert, gouvernance libre	Ouvert, gouvernance éditeur	Fermé, avec convertisseurs	Fermé

Le score individuel d'un composant logiciel est donc compris entre 0 (maîtrise totale du composant) et 21 (composant non maîtrisé).

4.8 Formules de calcul

Soit C_i un composant matériel ou logiciel. Le score individuel $S(C_i)$ de chaque composant est calculé sur la base des tableaux présentés dans les sections précédentes.

Les agrégats A_j sont construits de la façon suivante : l'agrégat d'un composant est constitué du composant et de l'ensemble des autres composants auxquels il est directement relié dans le graphe de dépendances. Un même composant peut donc être présent dans plusieurs agrégats. Par conséquent, les composants indépendants de tous autres sont placés dans des agrégats « individuels » qui ne contiennent qu'eux-mêmes.

Le score de dépendance de l'agrégat A_j est la somme des scores de tous les composants de l'agrégat. Il est donné par la formule suivante : $S(A_j) = \sum_{C_i \in A_j} S(C_i)$.

Chaque agrégat est pondéré par un coefficient de criticité, compris entre 1 et 3 (une valeur 0 indique que l'agrégat est situé hors du périmètre de l'étude), selon le tableau suivant. Le coefficient de criticité d'un agrégat A_j est noté $c(A_j)$.

Critère	Valeur			
	0	1	2	3
Pondération	Hors périmètre	Non critique	Moyennement critique	Critique

Le score global de dépendance $S(G)$ de l'organisation, c'est-à-dire le score calculé sur l'ensemble du graphe de dépendances G , est alors calculé selon la formule suivante :

$$S(G) = \max_{A_j \in G} (c(A_j) \times S(A_j)).$$

4.9 Interprétation

L'usage de la fonction « max » et de la pondération par la taille de l'agrégat permet de définir des objectifs atteignables. Cette formule incite à s'attaquer en priorité aux agrégats les plus massifs pour les dissocier, soit les plus critiques pour diminuer leur criticité, ce qui permet de faire baisser le score de l'agrégat considéré.

Pour autant, l'utilisation d'un « max » conduit à masquer la « dette de maîtrise », c'est-à-dire l'ensemble des efforts nécessaires pour faire diminuer les scores de l'ensemble des agrégats vers la valeur la plus basse possible. La dette de maîtrise peut s'exprimer comme la somme des scores des agrégats (incluant les scores individuels des composants non agrégés) :

$$D(G) = \sum_{A_j \in G} (w(A_j) \times S(A_j)).$$

5 Conclusion

Ce document présente une méthode permettant la prise en compte de la maîtrise des systèmes d'information dans la définition d'une feuille de route numérique. Cette méthode vise à la simplicité, pour pouvoir être mise en œuvre facilement et à moindre coût, et à l'opérationnalité, pour que les indicateurs qu'elle propose puissent se traduire en objectifs compréhensibles et atteignables.

Ce document est copiable et distribuable librement et gratuitement à la condition expresse que son contenu ne soit modifié en aucune façon, et en particulier que le nom de son auteur continue à y figurer, de même que le présent texte.

