Manifeste EIG (brouillon)

Sous-titre : Manuel d'utilisation des compétences numériques dans l'administration.

Qui sommes-nous ?

Nous sommes 28 designers, data scientistes, développeur·euse·s. Nous venons de la recherche, de start-ups, ou d'entreprise de toute taille. Nous avons des parcours d'ingénieurs ou de sciences humaines.

Nous sommes 28 citoyens, soucieux du bien commun. Notre goût pour le traitement de la donnée et les technologies du Web s'est mêlé tôt à notre sens civique : nous consacrons du temps à des projets de biens communs numériques et à des projets d'utilité publique.

Nous sommes 28 « Entrepreneur·e·s d'intérêt général » : recrutés par la mission Etalab de la DINSIC [Direction interministérielle du numérique et du système d'information et de communication de l'État], nous avons relevé pendant dix mois un défi au sein de l'État, en binôme ou en trinôme. Nous avons mis nos compétences humaines, techniques et créatives au service d'agents publics désireux de transformer leur administration et leurs outils numériques.

Au terme de nos expériences, nous avons fait la synthèse de principes que nous jugeons fondamentaux pour assurer le succès de la transition numérique de l'État, et nous souhaitons vous les partager.

À qui s'adresse ce manifeste ?

Ce manifeste s'adresse en priorité aux décideurs politiques et aux agents publics en charge de la transformation des politiques publics par le numérique. Plus largement, il vise tous les citoyens ayant à coeur, comme nous, d'améliorer le service public, notamment grâce aux nouveaux horizons du numérique.

Comment lire ce manifeste ?

Notre message s'articule en une série de constats et de recommandations. Avant d'y plonger, faisons un rappel essentiel.

Le numérique est un outil. C'est un outil phénoménal, de par la diversité des problèmes qu'il permet de résoudre, de par son efficacité, son évolution constante. Mais il ne sera jamais plus que cela : un outil. Comme tout outil, utilisé à mauvais escient, il peut s'avérer inutile, voire contre-productif, voire néfaste.

Le numérique n'est pas une baguette magique. Il n'est pas systématiquement le plus court chemin d'un problème à sa solution. Il ne coûte pas rien.

De même, les conseils que nous transmettons dans ce manifeste n'ont rien d'absolu. Ils ne remplaceront pas votre esprit critique, et ne s'adapteront pas à toutes les situations.

Notre espoir, en publiant ce document, est d'ouvrir un espace d'échange et de dialogue entre tous ceux qui se retrouvent dans nos constats, et qui sont curieux de mettre en œuvre les axes que nous proposons.

Notre vision

Nous estimons que les projets numériques les plus susceptibles de réussir sont ceux qui regroupent les caractéristiques suivantes :

Ils sont **de taille réduite**, utilisent des **composants les plus génériques possibles**, dans une approche par **amélioration continue**. S'appuyant sur des **ressources internalisées**, ils permettent de nouvelles formes de **mobilisation** et d'**ouverture**.

[Ordre suggéré pour les sections : réduire ; rendre générique ; améliorer ; internaliser ; mobiliser ; ouvrir.]

Idée générale

La taille des projets informatiques est inversement proportionnelle à leur chance d'aboutir. Or, il n'est pas rare de voir, dans l'administration, des projets informatiques de plusieurs dizaines de millions d'euros et la tendance est à l'augmentation de la taille de ces projets.

Diagnostic

En rajoutant des parties prenantes à un projet on ajoute mécaniquement une complexité managériale très coûteuse sur le long terme. La multiplication des interlocuteurs engendre un risque important de perdre le lien avec l' utilisateur final.

Plus un projet grandit, plus il devient compliqué d'être réactif, le temps entre chaque itération augmentant lui aussi pour de multiples raisons (coordination, sécurité, politique interne…). Il est alors plus difficile d'être certain de répondre au mieux aux besoins utilisateurs. Des frustrations internes apparaissent également puisque le temps nécessaire pour voir le résultat du travail accompli augmente. A l'heure où les technologies et les méthodes évoluent très vite dans l'informatique, en allongeant en parallèle les cycles de production on prend le risque d'utiliser des technologies et méthodes obsolètes compromettant ainsi l'avenir de la solution développée.

Enfin, un projet important qui aura nécessité des sommes conséquentes sera d'autant plus difficile à abandonner malgré des résultats médiocres constatés. Ce phénomène bien connu d'escalade de l'engagement, empêche l'analyse objective d'un projet et conduit à prendre de mauvaises décisions.

Proposition d'avenir : "Small is beautiful"

Ce n'est pas simple, mais casser un gros projet en de plus petit projets incrémentales fonctionnels et valorisables indépendamment.

Pratiquer le pre-mortem : imaginer toutes les raisons qui ferait qu'un projet ne fonctionnerait pas.

Exemple de projet bien délimité

Les musées doivent gérer des collections d'objets et les exposer via des sites web. Ces problématiques sont parfois abordées via un seul outil, servant à la fois à l'inventaire des objets et à la diffusion des données qui les concernent (images, descriptions, etc.)

Le défi EIG 2 du Mobilier national s'est concentré sur la conception et le développement d'une interface de diffusion et de valorisation. La problématique de la gestion des collections a été laissée de côté, et l'interface de diffusion a été conçue pour être paramétrable et se connecter à d'autres outils.

Les contre-parties de cette ambition limitée : les EIG ont exploré à fond la question des publics et ont proposé un outil que d'autres musées pourront réadapter à leurs besoins.

Les contre-parties de cette ambition limitée : les EIG ont exploré à fond la question des publics et ont proposé un outil que d'autres musées pourront réadapter à leurs besoins.

Idée générale

Le besoin de remplacement apparaît en général lorsque la dette technique est trop importante, que la solution n'a pas évolué depuis trop longtemps, que toutes les personnes qui ont travaillé au développement de la version actuelle ont changé de poste, et qu'on préfère repartir de zéro (aussi pour se dire que le nouveau chef apporte du changement… non du remplacement).

Se placer dans une logique d'amélioration demande de s'opposer à la logique statique de la souscription à un produit pour X années, produit souvent intégré par des prestataires, entraînant une forte dépendance technique et rendant difficiles les évolutions à venir.

Diagnostic

Proposition pour l'avenir

Nous recommandons une approche itérative et continue visant à délivrer une première version minimale du produit rapidement mais avec une vision ie en s'appuyant d'emblée sur la technologie pertinente pour les ambitions du produit. Par exemple si le produit vise une grande audience, il faut anticiper le passage à l'échelle en choisissant une architecture adaptée. Pour les perfectionnistes : ces contraintes ont un coût il est donc nécessaire de bien mesurer ses ambitions et ne pas choisir les technologies les plus robustes et les plus performantes PAR DEFAUT. En effet, ces technologies sont souvent les moins adaptées au développement rapide d'une "première version minimale du produit".

Exemples

Le site de la DREES

Le site web de la DREES est une solution web HTML4 qui s'appuie sur un framework web commercial. Par conséquent il est impossible de le faire évoluer librement, tout est facturé et les développement ne peuvent être effectués que par une seule entreprise. Aujourd'hui la DREES souhaite utiliser une autre solution pour exposer ces données afin d'exposer ses données de façon structurée et de gagner en référencement sur les moteurs de recherche. Afin de réaliser ce site, on privilégie des solutions open source et une ouverture du code du site afin de le rendre plus évolutif.

Enrichissement d'un outil

Dans la branche spécifique de la tarificaton en assurance, on calibre des modèles linéaires généralisés afin d'ajuster la prime aux risques. Pour cela, certains assureurs utilisent des outils commerciaux "clef en main" et qui fonctionnent en "clic-bouton". Aucune amélioration ou appropriation n'est possible. Le retour à des langages de programmation, par exemple R, a été douloureux pour les statisticiens qui ne savent plus coder. Néanmoins, lorsque je teste un outil alternatif (soit-il fermé ou ouvert), je découvre de nouvelles façon d'explorer, visualiser, modéliser les données.

- Si j'utilise une solution propriétaire et que je suis séduit par les fonctionnalités d'une autre solution. Je dois soit changer de solution soit utiliser les 2.
- Si j'utilise une solution open source et que je suis séduit par les fonctionnalités d'une autre solution. Je peux intégrer ces méthodes et idées, il n'est pas nécessaire de changer de solution, i.e. remplacer la solution actuelle. On peut implémenter la fonctionnalité ou créer une API entre un outil et un autre.

Idées générales

Créer des composants génériques bénéficie plus à la communauté que de construire une solution entière d'un seul coup, car votre besoin est très certainement celui de quelqu'un d'autre, demain ou aujourd'hui. D'ailleurs, votre problème d'aujourd'hui, quelqu'un l'a probablement rencontré auparavant et pourrait avoir beaucoup de choses à vous apprendre. Dans le meilleur des cas, un composant essentiel à votre solution existe même déjà.

Diagnostic / constat

Un projet avec une composante numérique part en général de l'expression d'un besoin métier précis : une cible d'utilisateurs ou encore des jeux de données sont déjà plus ou moins clairement identifiés, ainsi qu'une première idée des livrables. Il est souvent tentant - ou du moins plus confortable - de ne chercher à répondre qu'aux besoins premièrement exprimés sans penser à la possiblité que l'outil développé puisse être utilie ailleurs. Mais c'est souvent oublier que le problème que l'on rencontre dans son propre métier est ou a déjà été rencontré dans d'autres services, que cela soit dans sa propre administration ou non.

Oublier cet aspect c'est se couper dès l'abord d'un grand nombre de ressources potentielles : les personnes qui dans le passé ont déjà rencontré ce même problème pourraient apporter leurs propres retours d'expériences, leurs propres réflexions sur leur propre manière d'y faire face, les solutions précédemment développées pourraient être adaptées et faire gagner du temps dans le développement du projet, et ces personnes pourraient demain les mêmes qui seraient en mesure de contribuer et d'améliorer les solutions…

À l'inverse, aborder son propre projet en cherchant à le rendre le plus générique possible, et chercher à intégrer toutes les possibilités de personnalisations et d'adaptations dès le début risque d'alourdir le projet jusqu'à l'inefficacité. Il faudrait en effet identifier et rencontrer les potentiels réutilisateurs, caractériser leurs propres besoins, synthétiser les retours, et intégrer le tout dans la solution en cours… Les moyens à disposition de chacun étant limités, et les délais pour livrer étant eux aussi incompressibles, il est probablement impossible d'aboutir à des solutions 100% génériques.

La généricité d'un projet numérique est ainsi autant une question de méthodologie que de choix. Il est essentiel d'envisager les potentiels de réutilisation de sa solution afin de toucher une communauté plus large, mais il est tout aussi essentiel de pouvoir arbitrer entre les besoins métiers et leur effort de généralisation.

Proposition d'avenir

Nous proposons dans cette partie quelques principes permettant à des porteurs de projets numériques d'entamer une réflexion à la fois sur la portée générique de leur solution, mais aussi de leur permettre d'arbitrer dans leur gestion de projet.

Penser générique - en général

- Il faut penser dès l'abord aux réutilisations potentielles ou immédiates par d'autres de ce que l'on cherche à développer, ne serait-ce que pour se constituer une communauté.
- Il ne sert à rien de chercher une solution 100% générique, l'essentiel est de savoir ce qu'il est nécessaire de garder particulier, et ce qui est potentiellement généralisable.
- Il existe déjà un grand nombre de réponses totales ou partielles à son problème : cela va d'applications "clés en main" open source à des bibliothèques de script.
- Il faut relativiser et décomposer son problème en sous-parties, et alors identifier pour chaque partie ce qui correspond à un besoin métier, ce qui peut être repris et adapté, et ce qui doit être développé depuis zéro.
- De manière pragmatique il est plus simple de chercher à gagner en généralités à partir d'un élément précis du projet plutôt que sur l'ensemble.
- Mieux vaut bien faire des petits morceaux applicatifs génériques (c'est-à-dire une bonne documentation et une communauté d'utilisateurs), qu'une application complexe se voulant générique dans son ensemble.
- Se baser sur de l'open source : tous les projets informatiques actuels, y compris propriétaires, reposent sur des composantes libres. Créer des nouvelles briques libres est la continuité de ce mouvement.

Penser générique - une méthode

- Abstraire son propre besoin : le problématiser en tentant de faire ressortir les principes clés
  - exemple de besoin non généralisable : "mon service (la sous-direction du CGET à la politique de la ville) a besoin de montrer sur une carte de Passy les indicateurs de pauvreté des populations".
  - exemple de traduction générique du même besoin : "nous avons besoin d'un système permettant de récupérer des données dynamiques géolocalisées et de les afficher sur une carte en ligne"
- Parler de sa problématique à différentes personnes jusqu'à pouvoir expliquer son propre besoin de la même manière à tous.
- Décomposer la problématique générale en sous-parties :
  - besoins : besoins de l'équipe
  - données : d'où proviennent-elles, quelles opérations faire. Le modèle de données doit refléter l'usage générique qui en est fait.
  - interfaces : à qui s'adresse ce qui est développé,
  - contraintes : de temps, de moyens, de compétences…
- Délimiter ce que l'on cherche à développer de manière générique parmi l'ensemble des sous-problèmes et des fonctionnalités attendues. Une bonne approche à cette étape peut être de lister les fonctionnalités et de les classer par grandes familles.
- La conception de composants génériques se marie bien à une démarche de développement agile, itérative. Si certains composants peuvent être identifiés dès la phase de recherche initiale du problème métier, d'autres peuvent être extraits d'applications ou prototypes déjà développés.

Penser générique - des exemples plus appliqués

- Il est plus "simple" de chercher à généraliser des applications/fonctionnalités liées au traitement de données (ETL, nettoyage, agrégation), que des applications/fonctionnalités liées à une interface. En effet c'est au niveau des interfaces que se ressentent le plus vivement les besoins métiers spécifiques, alors que les opérations et automatisations autour des données en amont des interfaces sont plus "impersonnelles".
- Exposer des APIs (Application Programming Interface) aux bons endroits, ou encore se connecter à des API distantes est en général une stratégie payante à la condition de penser l'interopérabilité des données.
- Utiliser une DSL (Domain Specific Language) pour exprimer les différences métier d'un outil générique

Exemples concrets

TODO: choisir un ou plusieurs exemple(s) EIG ?

- tester une solution de prédiction sur une région dans l'idée de pouvoir la dupliquer à l'échelle nationale : PrévisSecours (repo)
- réussir à agréger des données publiées sur des pages html : OpenScraper (repo)
- développer des bibliothèques Python : TracFin (repo)

Idées générales

L'ouverture des algorithmes et des données est une obligation pour les aux administrations qui permet de XXX. Mais aussi :

- renforce leur légitimité par la médiation qu'elles mettent en place,
- renforce leur efficacité par les contributions extérieures et la réutilisation de leurs productions
- renforce leur utilité publique par une plus grande symétrie d'information.

Algorithme explicable, lisible, accessible, reproductible : permettre de comprendre les productions de façon claire, précise.

- médiation, ouverture code, ouverture données
- production de données, diffusion du savoir/ des ressources
- contributions extérieures (github/wikipédia/hackatons etc.), rayonnement

Construire, accompagner et diffuser des outils et méthodes explicites et réutilisables.

Concrétement

- logiciel libre
- ouverture des données
- interface lisible accessible
- transparence / symétrie / explicabilité

L’ouverture c’est de l’avenir - Ouverture vers le futur

- Favoriser
- Participer
- Augmenter
- Contribuer
- Transparence des algos, des usages, des origines
- Explicabilité
- Ouverture des contenus
- Diffusion
- Partage
- Réutilisation
- Accessibilité
- Médiation
- Compréhension
- Lisibilité
- Symétrie (le calculant et calculé puissent discuter de la mm chose)

Diagnostic (TODO)

Proposition d'avenir (TODO)

Exemple(s)

TODO: choisir soit un exemple parlant soit une série d'exemples.

- Atelier wikipedia au Mobilier national ?
- Logiciel libre / social connect
- Transparence :
    - J’informerai de façon compréhensible et précise toutes les parties prenantes sur les finalités, les modalités et les implications potentielles de mon utilisation des données.
    - Communiquer, ou sinon rappeler aux équipes compétentes, la nécessité de communiquer auprès des personnes concernées, l’usage qui sera fait de leurs données, de la façon la plus claire, explicite et transparente possible.

Idée générale

Aujourd'hui la plupart des projets logiciels au sein des administrations sont conçus de manière verticale. On conçoit un logiciel comme on conçoit un EPR: les objectifs sont trop ambitieux, le cahier des charges fait des milliers de ligne, les choix se font en vase clos avec peu de consultation des utilisateurs finaux, les processus sont lourds, les développeurs sont sous-valorisés et sous-staffés au profit de multiples couches de management intermédiaire. Résultat: les retards s'accumulent, la

facture augmente, les utilisateurs sont frustrés et lorsque la solution est enfin déployée, elle ne correspond plus aux besoins.

Au sein du programme EIG, nous pensons qu'il est possible de faire du logiciel autrement. Les entreprises du numérique ont montré la voie.

Diagnostic (TODO)

Proposition d'avenir (TODO)

Exemple (TODO)

Idée générale

Les outils numériques occupent le quotidien des agents publics et permettent à ces agents de réaliser leurs missions. Ces outils sont au coeur du travail et les agents ressentent le besoin de s'approprier les systèmes d'information. Ces utilisateurs ont besoin de faire évoluer leurs outils informatiques pour répondre aux besoins changeants de leurs missions et expriment le souhait de voir évoluer leurs outils informatiques plus facilement, sans d'importants délais et lourdeurs administratives.

Les attentes des utilisateurs concernant les outils numériques sont grandes, étant exposés à des logiciels de grande qualité dans leur usage du numérique dans la sphère privée. Ainsi, les DSI ont besoin d'adopter plus rapidement les technologies informatiques émergentes pour pouvoir proposer des outils attrayants. Dans la même idée, il apparaît comme souhaitable de conserver une relation de proximité entre les développeurs et le métier pour co-construire des outils pertinents pour l'utilisateur final. Enfin, la maîtrise des coûts reste un enjeu majeur pour le développement d'outils numériques.

Tout cela ne peut se faire qu'en internalisant plus de compétences informatiques, en décloisonnant les échanges entre DSI et métiers ou en s'assurant que des personnes en interne soient impliquées et comprennent en détail la partie du travail externalisée.

Situation passée

L'externalisation des projets informatiques est devenu de plus en plus courante au sein des administrations ces dernières années. La flexibilité de l'externalisation est en effet attrayante. Mais cette flexibilité cache plusieurs limites.

Le recours à des prestataires nécessite des appels d'offre qui prennent en moyenne plus d'un an juste pour la sélection d'un partenaire. Toute évolution des système est lente, fastidieuse et soumise à des lourdeurs administratives. Dans ces conditions, il est difficile pour les système d'information d'évoluer au même rythme que les nouveaux technologies et les évolutions métier.

Les DSI ayant un recours systématique aux prestataires accumulent une dette technique : ils n'ont plus la main de leur propre SI. Cette dette entraîne une relation de dépendance malsaine avec un acteur extérieur dont l'objectif est de maximiser ses revenus. Dans les faits, la maintenance et les évolutions du projet sont souvent sous-estimé.

Pendant la réalisation d'un projet informatique, une grande partie de la valeur créée l'est dans les cerveaux de ceux qui le réalisent. La compréhension du métier, des besoins, des systèmes et des données ont une valeur très précieuse bien que difficilement quantifiable. Quand les prestataires finissent le projet, toute cette valeur quitte l'administration.

Proposition pour l'avenir

Les besoins SI changent fréquemment et nécessitent l'intégration constante de nouvelles connaissances et technologies. Une réinternalisation des compétences informatiques permet de piloter de façon plus efficace le système informatique de l'administration au sens large. Cela permet une meilleure flexibilité qui évite de coûteuses renégociations ou des difficultés d'intégration de ces nouvelles connaissances par des prestataires externes. Par ailleurs, les coûts liés à l'interopérabilité des différents outils informatiques utilisés sont considérablement diminués. Cette réinternalisation doit être accompagnée par une montée en compétence du personnel interne ainsi que par le recrutement de nouveaux talents, notamment des développeurs, des data scientists et des designers dans les DSI.

Le programme EIG permet à la fois de démontrer la viabilité d'un réinternalisation ainsi que d'intégrer de façon temporaire dans l'administration des professionnels capables de faire monter en compétence les développeurs internes. Cela permet aussi d'exposer les DSI à des pratiques du monde entrepreneurial. Le programme donne l'occasion d'échanger entre services métiers et DSI pour la conception des outils ainsi que de mutualiser les méthodes, outils et les pratiques avec d'autres administrations.

Exemples

Internaliser ne veut pas dire que tout est réalisé en interne. Cela signifie qu'il y a au moins une personne agissant au nom de l'administration à même de maîtriser les enjeux technologiques du métier.

Dans le cadre du programme EIG, plusieurs défis ont organisé des formations (R pour Lab Santé et SQL pour Prédisauvetage), expérimenté des technologies émergentes (VueJS pour SocialConnect et Prédisecours, Docker pour Hopkins, ReactJS pour Archifiltre et Swagger pour Brigade Numérique) et ont sensibilisé l'administration à des méthodes de conduite de projet (l'agile pour ArchiFiltre) ou à l'importance du design pour Brigade Numérique.

Conclusion

"Ça ne peut pas se faire du jour au lendemain !" "Non, mais on peut avoir commencé, du jour au lendemain."

- Isaac Asimov

"La phrase la plus excitante à entendre en science, celle qui annonce de nouvelles découvertes, n'est pas «Eureka» (j'ai trouvé!), mais plutôt «Tiens, c'est marrant…»"

- Isaac Asimov

"N'interrompez jamais quelqu'un qui est en train de faire quelque chose que vous aviez dite impossible."

- Amelia Earhart

Qui nous sommes (un paragraphe)

Nous sommes une trentaine de profils de techniciens (datascientistes, développeurs, designers). Nous avons tous eu une expérience professionnelle hors de l'administration. Le programme EIG nous a permis de participer à un projet informatique au sein de l'administration publique pendant dix mois. Nous avons croisé nos constats, identifié plusieurs problèmes récurrents et nous

proposons une série de six principes qui ont vocation à guider les décideurs et les agents dans la définition et la mise en oeuvre d'un projet informatique.

Articulation des principes

Les principes vont par deux sur trois axes :

- Les principes qui ont surtout trait à la gestion des ressources humaines : **faire soi-même** (recruter, former, décloisonner) et **mobiliser** (impliquer les parties prenantes dans une gouvernance horizontale réactive).
- Les principes qui portent sur la « vision produit » (le niveau stratégique) : **limiter** (réduire le périmètre fonctionnel, la durée et donc les risques) et **améliorer** (privilégier des cycles courts, un maintien continu et une évolution constante).
- Les principes qui portent sur les aspects techniques : **composer** (créer des services interopérables pour gagner en souplesse) et **ouvrir** partager le code source pour mieux communiquer.

On peut imager ça avec un « hexagone vertueux » :

Faire → Limiter → Composer → Mobiliser → Améliorer → Ouvrir

Ça pourrait se lire ainsi :

- Faire soi-même incite à se limiter
- Se limiter implique de penser en termes de briques composables
- Des briques composables sont propices à la mobilisation
- La mobilisation permet une logique d'amélioration continue
- L'amélioration continue "s'augmente" avec l'ouverture

C'est forcément un peu arbitraire, mais je crois que le résumé doit proposer une articulation, d'une façon ou d'une autre, montrant que tout se tient.

Questions clefs

L'administration publique est responsable de ses outils informatiques. Elle en assure la maîtrise, la continuité et la pérennité. Voici une liste de questions pour motiver le recours aux principes que nous proposons.

- Mon service peut-il prendre en charge le développement d'un outil indispensable à mon métier au lieu de le confier à un tiers ?
- Le périmètre de l'outil que nous développons a-t-il été réduit au strict nécessaire ?
- L'outil que nous développons a-t-il été conçu pour s'interfacer avec des outils existants et pour exposer des données structurées ?
- Tous les profils d'utilisateurs de l'outil ont-ils été impliqués dans sa conception ? Sont-ils impliqués dans des tests continus ?
- Le code source de l'outil que nous développons est il ouvert ? Publions-nous une feuille de route et guidons-nous les contributeurs potentiels vers les contributions qui nous importent ?

Ces questions peuvent être posées par tout le monde à tout moment.

Améliorer

- https://www.stonebranch.com/replace-and-enhance/

  Commentaire : Replace & Enhance sont incompatibles, selon cette SSII https://www.stonebranch.com/replace-and-enhance/ on peut acheter un service pour remplacer OU améliorer notre solution actuelle, dans ce cas améliorer sous-entend ajouter une nouvelle solution en parallèle de la solution actuelle, ce qui va seulement ajouter de la complexité.
- https://www.evidenceinmotion.com/blog/2017/03/26/repeal-and-replace-vs-tweak-and-improve/
- Affordable Care Act (ACA) => So our only viable option is the ACA improved (Medicare has been improved hundreds of times since 1965) with critical analysis of it's coverage of all
- Est-ce qu'un algorithme doit automatiser ie remplacer l'intervention humaine ou l'améliorer avec des outils d'aide à la décision ? https://www.brookings.edu/blog/education-plus-development/2018/06/11/the-new-humanism-technology-should-enhance-not-replace-human-interactions/
- Le risque d'un "outil d'aide à la décision" c'est qu'il recommande une solution et que l'intervention humaine consiste simplement à valider, systématiquement, la première recommandation.

Limiter

- Small Is Beautiful: A Study of Economics As If People Mattered, Schumacher (wikipédia)

Faire soi-même

À faire

- Recruter des développeurs et datascientistes et designers dans les DSI
- Faire monter en compétences les développeurs existants
- Amener des gens capables de faire monter en compétence les développeurs présents
- Créer plus d'occasions d'échange entre services métiers et DSI pour la conception des outils
- Créer plus d'occasions d'échanges entre gens de mêmes métiers d'autres administrations
- Permettre les expérimentations dev. mais les encadrer pour éviter les fausses promesses, les outils de doublon
- Accepter l'innovation incrémentale, la techno parfaite n'existe pas
- Exposer les DSI à des pratiques du monde entrepreneurial

À ne pas faire

- Créer de la frustration en n'encadrant pas les expérimentations
- Placer la barre trop haut (ou au mauvais endroit) sur les exigences de qualité avant mise en production
- Développer des outils « mammouths »

- Développer sans prendre en compte les besoins en ergonomie des utilisateurs

Mobiliser

À faire

- Impliquer les utilisateurs: recueillir des besoins, élaborer des personas, faire preuve d'empathie
- Inviter aux contributions grâce à:
  - code ouvert
  - suivi de projet (eg Kanboard) publique
  - Issue tracker, possibilité de proposer des améliorations
- Valoriser les carrières des "doers" sans forcément les promouvoir manager. Nous avons besoin de profils techniques avec de l'expérience.
- Respecter le rythme de travail des "doers", Cf Paul Grahm => http://www.paulgraham.com/makersschedule.html.
- Avancer de manière incrémentale. Publier un MVP le plus rapidement possible pour créer une dynamique. Déployer le produit sur un maximum de postes utilisateurs dès le début.
- Communiquer le plus possible en faisant des démos
- Redevabilié et responsabilité d'une équipe sur un projet versus on se rejette la faute les uns sur les autres.

À ne pas faire

- Rédiger un gros cahier des charges fixe en vase clos entre décideurs.
- Multiplier les couches de management intermédiaire entre la personne qui produit le code et la personne qui consomme le logiciel.
  - par ex. Développeur sous traitant > Manager sous traitant > Chef de projet Maitrise d'oeuvre > Référant utilisateurs > Utilisateurs
- Multiplier les longues réunions. Il est préférable d'organiser des stand-up assez courts