CFTL Comité Français des Tests Logiciels

ÉVOLUTION DES TESTS LOGICIELS

Ouvrage collectif coordonné par :
Olivier Denoo
Marc Hage Chahine
Bruno Legeard
Eric Riou du Cosquer

Edité par le Comité Français des Tests Logiciels

ÉVOLUTION DES TESTS LOGICIELS

Ouvrage collectif coordonné par :
Olivier Denoo
Marc Hage Chahine
Bruno Legeard
Eric Riou du Cosquer

Edité par le Comité Français des Tests Logiciels

SOMMAIRE

PRÉFACE

par Olivier Denoo, président du CFTL

Notre société est en perpétuel changement. Elle redessine ses contours à un rythme de plus en plus rapide. Elle devient impatiente.

C'est le monde du « tout, tout de suite »… parce que la vie est courte et qu'on ne vit qu'une fois. C'est un monde de plus en plus numérique - du moins pour ceux qui peuvent se le permettre - où les technologies de l'information s'invitent partout, jusque dans les moindres recoins de nos vies et de nos objets.
Abreuvé par cette véritable hydre technologique, il se complexifie, digérant à un rythme effréné les produits et services qu'il intègre ou qu'il fait communiquer au gré de ses humeurs ou de ses besoins. Il devient mobile, nomade, hyperconnecté, transcendant ainsi les cultures, les organisations et les frontières.
Il devient portable, modulaire : c'est l'ère des systèmes d'information hybrides, des objets connectés, de l'Intelligence Artificielle, du passage à l'échelle, du tout continu et des micro-services. Il se globalise et nous rapproche, dans une certaine mesure… car cet univers digital connaît aussi ses défis, ses échecs, ses cimetières technologiques et ses laissés-pour-compte qui font du stop sur le bas-côté des autoroutes de l'information.

Loin de n'être plus qu'une conception des informaticiens, pour des informaticiens, nos technologies de l'information se confrontent de plus en plus à la « vraie vie », celle de tout un chacun, avec ses choix, ses doutes et ses hésitations.
Avec ces « nouveaux » utilisateurs, novices, peu soucieux de vouloir comprendre ce qu'il y a sous le capot mais qui ont un avis sur tout, de nouvelles valeurs s'imposent peu à peu et viennent bouleverser encore cette mosaïque complexe : utilisabilité, accessibilité, sécurité, vie privée, écologie et green IT, Lean IT, éthique et inclusion…

Pour répondre à ces différents challenges, les organisations ont dû adopter de nouvelles approches, plus souples, plus Agiles. Des approches qui embrassent le changement, acceptent l'erreur – surtout si elle survient vite - en se concentrant sur la valeur. Tout cela sans lever le pied une seule seconde car il faut être le premier ou le meilleur et que le temps c'est de l'argent !

Voilà donc qu'il nous faut repenser l'organisation et la chaîne de valeurs, voilà qu'il nous faut raccourcir les cycles, rapprocher les « extrêmes » pour plus d'efficacité opérationnelle.
Alors nous automatisons à tout crin pour remettre la pyramide de test à l'endroit, nous développons les approches en « DD », nous devenons DevOps ou DevsecOps, nous mettons le focus sur la chaîne de valeurs pour mieux la délivrer itérativement et en continu. Soumis à une véritable quadrature du cercle, nous tentons de simplifier les tâches, de complémenter les rôles et les compétences au travers d'équipes cross-fonctionnelles, nous faisons plus de code, mais en mode « codeless ».

De nouveaux modes de gestion, de prises de décision et de communication voient peu à peu

le jour, avec plus ou moins de bonheur. Le « shift-left » devient (enfin ?) une réalité, tandis que le test en production, autrefois considéré comme un tabou absolu, devient monnaie courante maintenant qu'il s'appelle « shift right ».

Nos structures traditionnelles éclatent, les services ou les projets disparaissent pour le plus grand profit des Teams, Squads et autres Tribes pour laisser gérer nos développements avec une très large autonomie. Les fonctions et les rôles deviennent des tâches et, souvent, ce qui était l'affaire d'un seul devient l'affaire de tous… et parfois, hélas, de personne !
Ces remaniements se font parfois au risque d'y perdre notre latin, nos repères, une certaine cohérence. Le risque de voir la qualité dispersée, diluée, minimisée voire marginalisée, est bien réel au sein de nos organisations faites d'équipes pluridisciplinaires, indépendantes les unes des autres… alors que cette qualité devrait, au contraire, rester un différenciateur majeur, garant de la satisfaction de nos utilisateurs.

Le testeur n'est bien évidemment pas épargné par ces changements structurels et structurants. Tantôt son rôle se renforce : il devient coach qualité ; il fait du conseil, de l'assistance qualité ou du Q.E. (« quality engineering ») ; il se rapproche du métier, du P.O. et du développeur ; il touche à tout, des exigences à l'organisation, et transcende son art et son savoir-faire.
Tantôt son rôle se dilue, s'amoindrit : il se perd dans la masse, devient un technicien parmi les autres et peine à faire valoir son avis. Il doit se refaire une petite place dans cette part de la culture digitale 3.0 qui n'a rien compris à l'Agilité et à ses préceptes ; elle qui cède si aisément à la facilité sans se rendre compte qu'elle va plein gaz… en marche arrière.
Le testeur qui a tant peiné à se faire reconnaître attend à nouveau que son heure sonne.

Le tableau ne serait pas complet sans l'adjonction progressive d'outils et techniques basés sur l'IA, ce nouveau Graal technologique censé nous surpasser, à terme, en (presque) tout (ou non).
Même si elle n'en est encore qu'à ses balbutiements, l'IA est déjà présente dans de nombreuses activités de test et dans l'imaginaire de nombreux testeurs. Que ce soit pour créer des cas de test à partir de l'analyse des logs et des usages, aider au reporting ou à la prédiction des défauts, détecter les changements d'interface ou assigner les défauts à la bonne personne, il y a déjà une IA pour répondre à l'appel.

Pourtant, alors que la plupart des approches actuelles font la part belle à la collaboration, il est regrettable de constater que les aspects humains qui en découlent restent encore aujourd'hui un domaine largement inexploité dans les cursus universitaires ou les certifications professionnelles.
Que l'on officie au sein d'une équipe ou d'une tribu, que l'on soit membre d'un Squad ou d'une Guilde, la problématique reste la même : il faut communiquer !

Communiquer est certes à la portée de (presque) tout le monde… mais communiquer efficacement, se faire comprendre par l'autre, sans ambigüité et sans détour, harmoniser les termes, les usages et méthodes pour éviter de monter de nouvelles tours de Babel, c'est une autre paire de manches ! En témoignent les cadavres exquis[1] et les intemporels 40% d'exigences peu claires à l'origine de bien des déboires dans nos projets.

[1] Le cadavre exquis est un jeu collectif inventé par les surréalistes et notamment par Jacques Prévert, en 1925. C'est un jeu qui consiste à faire composer une phrase, ou un dessin, par plusieurs personnes sans qu'aucune d'elles ne puisse tenir compte de la collaboration ou des collaborations précédentes. C'est ce qui rend le jeu du cadavre exquis si amusant, car le résultat obtenu n'a souvent ni queue ni tête. L'une de ces séances aboutit à la phrase restée célèbre : « le cadavre exquis boira le vin nouveau ».

Qu'il s'agisse d'un sport collectif ou d'un projet où l'équipe dispose d'une plus ou moins large autonomie, la magie n'opère que dans la complémentarité des rôles. Aucun entraîneur de football qui se respecte n'irait composer sa « dream team » avec uniquement des attaquants ou des gardiens de but, aucun entraîneur de rugby ne ferait fortune avec seulement des demis de mêlée…
De même, une équipe informatique doit s'appuyer sur des compétences multiples et complémentaires, des compétences dont on n'a pas forcément besoin à tout moment, mais qui font cruellement défaut quand elles manquent à l'appel. Un visionnaire n'a pas toujours le souci du détail et peut manquer d'empathie, ou détester l'organisation et les tâches administratives qui l'accompagnent. Tout le monde n'est pas fait pour gérer des équipes ou regarder le monde avec l'esprit critique. Chaque tâche, chaque rôle, requiert des compétences spécifiques pour lesquelles nous développons, au cours de notre vie et de nos expériences, plus ou moins d'appétence.
Au sein de l'équipe, chacun apporte sa petite pierre à l'édifice commun, en fonction de ce qu'il sait faire mais aussi de ce qu'il aime faire (car en général cela va tellement mieux quand on aime ce que l'on fait).

Si nos entreprises veulent réussir le défi de la transformation digitale, elles devront immanquablement s'appuyer sur les meilleures pratiques, tant économiques que pour la qualité.
Si notre enseignement, universitaire, privé ou professionnel veut rester compétitif et pertinent, il faudra qu'il intègre l'humain en tant que composant indissociable au cœur de la technologie.
C'est aussi le rôle et le défi des standards comme l'ISTQB® qui, - s'ils harmonisent déjà les pratiques et le vocabulaire, s'ils inspirent et guident les équipes, doivent aussi définir les outils et méthodes qui permettent de garder le contrôle, de structurer et de piloter les livraisons tout au long de la chaîne de valeurs.

L'ouvrage que vous tenez entre les mains est le fruit d'un travail collectif, coordonné par le CFTL et alimenté par les réflexions de nombreux experts, chacun dans son domaine respectif.
Il tente, modestement, de vous apporter des pistes de réflexion mais aussi des réponses aux questions que vous vous posez, dans ce monde qui change et qui nous déstabilise (n'est-ce pas là la définition de « disruptif » d'ailleurs ?).
Nous avons choisi de couvrir de nombreux sujets d'actualité (IA, GreenIT, RGPD…), mais aussi des sujets plus classiques ou récurrents, parce qu'ils sont eux aussi impactés par la marche de la transformation digitale et bientôt, qui sait, du Metaverse et de la réalité virtuelle au quotidien.
Nous espérons qu'il vous sera à la fois utile et agréable.

J'en profite enfin pour remercier chaleureusement l'ensemble des auteurs et des coordinateurs pour leur immense contribution, sans laquelle ce livre n'aurait pu voir le jour.

Présentation du CFTL et de l'ISTQB®

par Olivier Denoo, Bruno Legeard et Eric Riou du Cosquer

Fondée en 2002, l'association internationale type loi 1901 (association à but non lucratif) ISTQB® – International Software Testing Qualification Board – a permis de formaliser le corpus de compétence des métiers et rôles des tests logiciels.

La reconnaissance des compétences individuelles et la création de parcours de compétences sont des éléments clés pour la mise en place de parcours professionnels attractifs. La gestion des carrières dans le domaine des tests logiciels est facilitée par le schéma ISTQB® qui, à partir du socle Testeur Certifié Niveau Fondation, intègre les modules Agile (Fondation et Testeur Technique Agile de Niveau avancé), les modules spécialisés (Acceptance Testing, Model-Based Testing, Tests d'IA, Utilisabilité, Performance…) ,les niveaux avancés (Test Manager, Analyste de Test, Analyste Technique de Test) et les niveaux experts (Assessing the test process, Implementing Test Process Improvement).

Comme le montrent les différentes enquêtes de l'ISTQB®, la motivation principale des employeurs et des professionnels dans le passage des certifications se trouve dans le développement et la gestion des compétences et dans une vision de long terme de leur renforcement et de leur certification.

En quelques chiffres, l'ISTQB®, c'est :

- Plus de 100 pays membres sur tous les continents.
- En 2023, le monde comptait près d'un million de professionnels certifiés par l'ISTQB®.
- Une croissance soutenue pour la certification Testeur Agile et les modules spécialistes.
- Un partenariat fort avec le schéma IREB® – International Requirements Engineering Board – sur l'ingénierie des exigences, TMMi (Test Maturity Model integration) et IQBBA (International Qualification Board for Business Analysts).

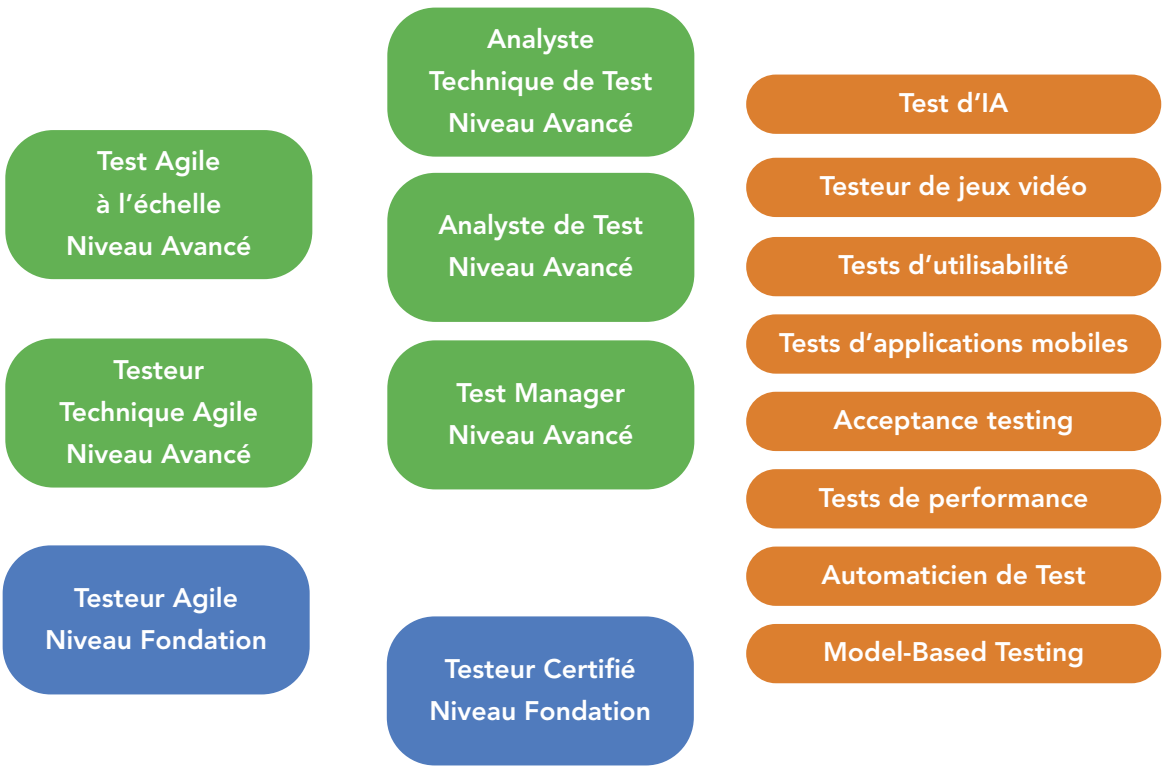
L'ISTQB® poursuit des efforts constants afin de développer et de mettre à jour ses différentes certifications, ainsi que son Glossaire des termes des tests logiciels, en phase avec l'évolution des pratiques, dont celles issues de l'Agilité, du monde mobile ou de l'Intelligence Artificielle.

Le CFTL – Comité Français des Tests Logiciels – est le représentant officiel de l'ISTQB® en France, organisant ainsi à la fois la promotion des certifications ISTQB®, l'accréditation des organismes de formation et la traduction française de l'ensemble des matériels. Le CFTL participe aussi directement aux groupes de travail internationaux de l'ISTQB®.

Le saviez-vous ?

Avec plus de 1 200 000 examens cumulés passés dans le monde en 2023, les certifications ISTQB® comptent parmi les plus diffusées des technologies de l'information.

Certifications CFTL/ISTQB® disponibles en France en 2024
https://cftl.fr/



Depuis 2008, le CFTL organise la Journée Française des Tests Logiciels qui accueille, cette année encore, plus de 1000 participants et 40 exposants.

PARTIE I
POUR UNE QUALITÉ DURABLE


TU TOURNES LE DOS AU LOGICIEL POUR LE TESTER ?
OUI... JE TESTE D'ABORD SI JE PEUX M'EN PASSER.
ET SI C'EST LE CAS, JE LE JETTE AU NOM DE LA FRUGALITÉ...
LOGICIEL
CLIQUEZ LÀ
OU LÀ
OU ICI
OK ?
A. COMTE

I.0- Définition de la qualité durable

par Marc Hage Chahine

Nous sommes dans un monde qui subit de grands changements. Ces changements ont de nombreux impacts, dont certains sont particulièrement préoccupants. Je pense notamment aux problèmes écologiques, au désengagement des employés (comme avec la grande démission aux USA) ou encore à une société qui semble toujours plus fracturée.

L'industrie du logiciel, même si elle n'est pas responsable de ces changements, contribue, à son échelle, à ces derniers. Il est nécessaire que les services numériques s'adaptent à ces changements, ou nous ne pourrons plus évoluer dans un monde viable tant au niveau environnemental qu'au niveau sociétal.

Afin de répondre à ces défis actuels, il est nécessaire de penser la qualité autrement et d'aller vers une qualité durable. Qu'on se le dise une bonne fois pour toutes, la vraie qualité ne peut être que durable.

Mais, au final, qu'est-ce que la qualité durable ?

1- Les valeurs de la qualité durable

La qualité durable, c'est avant tout un état d'esprit. Un état d'esprit qui repose sur des valeurs :

- La sobriété matérielle

La construction représente plus de 75% de l'impact écologique d'un smartphone. Il est essentiel, pour l'environnement, de proposer des services qui fonctionnent sur des terminaux anciens, afin de limiter le besoin de changement pour accéder à ces services. Au-delà de l'écologie, une sobriété matérielle est également importante pour la société et l'accessibilité aux personnes à faible revenus.

- La frugalité du logiciel – ou logiciel juste

Les logiciels sont de plus en plus gros et complexes. Mais en a-t-on vraiment besoin ? La réponse est non. En moyenne, près de la moitié (45%) des fonctionnalités d'un logiciel ne sont jamais utilisées et seulement 20% utilisées fréquemment ou toujours (source conférence XP 2002 de Jim Johnson) ! L'Agile prône cette simplicité et ce logiciel « juste », épuré de ces excès comme le rappel ce principe : « La simplicité – l'art de maximiser la quantité de travail qu'on ne fait pas – est essentielle ».

- L'humain

Les services numériques sont des outils qui doivent rester au service des humains et des utilisateurs. Tout logiciel doit se concentrer sur ces éléments. Cela inclut l'abandon des abus liés à l'économie

de l'attention, l'exploitation des données mais aussi la prise en compte des utilisateurs dans toute leur diversité.

De même, il est nécessaire de penser aux créateurs des services numériques et de s'assurer de leur bien-être et de leur engagement.

- Le souhaitable

L'idée est de toujours penser aux avantages et inconvénients du service à court, moyen et long termes, afin d'évaluer si cela vaut vraiment le coup de proposer ce service.

Cela force à évaluer les impacts positifs et négatifs (sur l'environnement et les personnes), à penser aux usages probables mais aussi aux alternatives possibles. Evaluer l'ensemble de ces éléments permet de définir si oui ou non, on a plus ou moins intérêt à avoir ce service et donc à souhaiter son « existence ». Car oui, parfois il est préférable de ne pas faire !

2- Les piliers de la qualité durable

Vous l'aurez compris, les valeurs de la qualité durable sont essentielles pour répondre aux problématiques environnementales d'abus liés aux services numériques, mais aussi au mal-être que l'on peut observer dans les équipes contribuant à la construction des logiciels.

C'est d'ailleurs en partant de ces piliers que j'ai commencé à travailler sur l'idée de cette qualité durable :

3- Définition de la qualité durable

Au final, il reste compliqué de bien définir la qualité durable. Si je devais vraiment m'engager sur une définition, actuellement, cela serait :
La qualité durable est le fait de proposer un service numérique de qualité sur le long terme du point de vue environnemental, des créateurs et de la société.

I.1- Le numérique responsable, pourquoi ? Comment ? Quel rapport avec les ours polaires ?

par Lydie Huon et Manaëlle Perchet

1- Les impacts du numérique sur l'environnement.

Au lendemain de la Pandémie du Covid 19, le monde n'a jamais été aussi enclin à construire "un monde nouveau", plus conscient, plus responsable et davantage aligné avec les enjeux de notre siècle: assurer la pérennité de la vie sur Terre en respectant les principes du développement durable, à savoir être économiquement viable, avoir un impact positif sur la société et dans le respect de l'environnement.
En ce sens, le numérique a une carte à jouer. Mais laquelle ?

De la montre connectée à la prise de rendez-vous en ligne, en passant par le tracking de nos livraisons et la géolocalisation de nos chérubins, le numérique est partout, tout le temps. Quinze fois plus d'appareils numériques que d'habitants, c'est la tendance que l'on observe dans les pays occidentaux, mais cette surconsommation s'applique à l'ensemble de la planète. Difficile de s'en passer…

Alors, le numérique: problème ou solution ?
En réalité, le poids de la consommation numérique s'étend bien au-delà de la consommation d'électricité. Il s'agit avant tout d'un impact sans précédent sur la consommation d'énergie primaire, d'émission de gaz à effet de serre et de l'épuisement des ressources abiotiques, du pétrole et des métaux critiques nécessaires à la fabrication de nos smartphones et tablettes. Cette progression exponentielle met en danger la disponibilité de ces ressources, ces dernières n'existant pas (par définition) en quantité illimitée. Un chiffre clé : d'ici 2025, nous pouvons nous attendre à ce que l'empreinte environnementale du numérique passe de 2 à 6%, soit une multiplication par 3 de son empreinte à l'échelle mondiale par rapport à 2010.

En France, le numérique représente déjà 6,2% de la consommation d'énergies primaires, soit 3,2% des émissions de gaz à effet de serre du pays, l'équivalent de 2,2% de la consommation d'eau, ou encore de 4 milliards d'excavation de tonnes de terre. Un digital pas si Immatériel n'est-ce pas ?

Le risque principal d'une telle trajectoire résiderait donc dans la raréfaction progressive de ces ressources. Dans ce cas, nous pourrions nous attendre à une hausse des prix des produits finis, permettant aux plus riches de continuer de bénéficier des externalités positives du digital comme la culture, le divertissement, les soins ou encore l'enseignement, mais éloignant les "laissés pour compte" du numérique : ceux pour qui le digital deviendrait inaccessible, tant en terme d'usage que de prix.

Le rôle des entreprises est donc crucial : tant dans la démocratisation des outils et services proposés par le digital, que dans l'innovation technologique durable, à travers le développement

de la sobriété numérique, de l'écoconception et de l'accessibilité. Autrement dit : donner les moyens et les outils permettant de consommer mieux, donc de consommer moins, pour un futur plus durable.

Malheureusement, les entreprises tardent le plus souvent à prendre leurs responsabilités : soit à cause d'un manque de "sentiment d'urgence" de faire évoluer leurs pratiques, soit simplement par manque de moyens opérationnels et de connaissances en matière de développement et de sensibilisation. Pourtant, le potentiel est grand et les opportunités multiples. Cela permettrait notamment de nourrir la triple performance de l'entreprise décrite par John Elkington dès 1998, par l'évaluation des critères de performance sur les trois piliers de développement durable : People, Planet, Profit.

Mais alors, le numérique est-il simplement une source de problèmes ? Ne pourrait-il pas faire partie de la solution ?
Parce que le numérique est un formidable levier pour la transition écologique et la lutte contre le changement climatique, nous souhaitons que d'ici 2025, il soit un moyen pour participer à la transition et non pas un outil qui contribue toujours davantage à la hausse des émissions.

2- L'obésiciel : lorsque nos logiciels deviennent obèses…

Microsoft 365 requiert 171 fois plus de ressources qu'Office 2010 et le web en 2023, c'est 155 fois plus d'énergie qu'en 1995 d'après Frédéric Bordage (fondateur greenIT.fr).
Ces chiffres vertigineux illustrent une terrible réalité. A travers l'évolution des technologies, nous arrivons à générer de formidables innovations : miniaturiser et mutualiser les appareils (ex: smartphone pour téléphoner, naviguer en ligne, regarder des vidéos, prendre des photos, écouter des podcasts), à faciliter le lien social (garder contact malgré la distance, maintenir le contact pendant les confinements), à faire de nouvelles avancées scientifiques en faveur de la santé, ... Nous arrivons plus facilement à mettre en œuvre des concepts sophistiqués, notamment grâce aux performances de nos machines. En revanche, nous ne nous questionnons plus sur les ressources utilisées ni sur leur quantité (nécessaires notamment pour la fabrication des appareils), la mémoire nécessaire, la consommation d'énergie, … Par exemple, les 70 Ko qui nous ont permis d'aller sur la Lune en 1969 suffisent à peine à envoyer un mail aujourd'hui.
Nous sommes tous équipés de machines qui rament sous le poids des mises à jour successives d'applications que l'on utilise peu ou pas.
Il existe plusieurs raisons à cela. En voici deux qui nous semblent importantes :

- Les fonctionnalités "au cas où"

Lorsque les produits logiciels étaient délivrés en une fois (ex : gravés sur CD/ DVD) on prévoyait une multitude de fonctionnalités additionnelles en essayant d'anticiper un maximum de besoins utilisateurs. Il s'agit d'une tendance qui persiste et qui induit, par conséquent, certains automatismes poussant à " spécifier au maximum" ! Car une fois que ça part en développement, on ne pourra plus modifier / ajouter" (wink le cycle en V !).

Vous avez peut-être entendu parler de **la loi de Pareto** (principe du 80-20 pour les intimes) : 20% de causes permettent de résoudre 80% du problème, autrement dit 20% du périmètre portent 80% de la valeur. La priorisation par la valeur repose donc sur ce principe qui consiste à réaliser au plus tôt ce qui va apporter le plus de valeur à l'utilisateur, afin de ne pas gâcher de l'énergie (humaine, matérielle et financière) sur des fonctionnalités qui ne seront que peu (voire pas) exploitées.

Les fonctionnalités "au cas où" ont un effet dévastateur : selon le collectif GreenIT.fr, 45% des fonctionnalités des services numériques ne sont jamais utilisées et 25% le sont rarement.
Ceci ayant des conséquences multiples:
1) Sur la pertinence du projet : seulement 29% des projets IT sont livrés OTOBOQ (On time, on budget, on quality).
2) Sur l'expérience utilisateur : le poids des pages web a été multiplié par 115 en 20 ans.
3) Sur les surcoûts matériel : notamment liés aux équipements et à l'hébergement.
4) Mais également des surcoûts de main d'œuvre : puisque 70% des coûts d'un service sont liés à la maintenance de ce dernier.

- Les couches d'abstractions logicielles

L'évolution des langages de programmation facilite grandement la conception de logiciels, avec des effets de bord pas toujours désirables.
Plus un langage est évolué, plus les couches d'abstractions vont se succéder pour faciliter la vie du développeur. Par exemple, la gestion mémoire en langage C, MALLOC-CALLOC (cauchemar de beaucoup d'étudiants) est gentiment occultée par le fabuleux garbage collector java qui fait tout ce boulot pour nous…
Plus le langage est sophistiqué et plus les couches d'abstractions vont être nombreuses.

Le progrès technologique (miniaturisation des transistors par exemple) repousse les limites et offre la possibilité d'exploiter des langages plus complexes mais aussi plus consommateurs en ressources mémoire et énergie. Cela facilite nos quotidiens mais à quel prix ? On va aller chercher les fonctionnalités sophistiquées dans des librairies de plus en plus grosses, ce qui va alourdir le logiciel, vieillir prématurément les appareils des utilisateurs en consommant de la mémoire vive et du stockage superflu. Les infrastructures d'hébergement et de transfert vont également être impactées par l'énergie consommée et le volume de stockage que représente cette librairie supplémentaire. La facilité a un coût qu'il est urgent de questionner, faire un usage raisonné des librairies disponibles devient notre responsabilité en tant que professionnels du logiciel.

3- Et nos utilisateurs dans tout ça ?

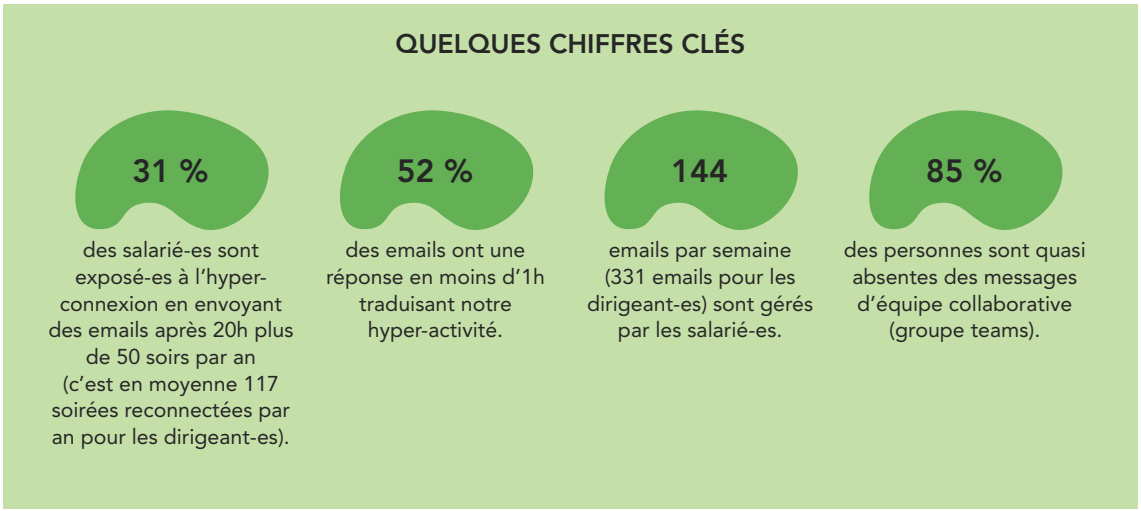
Les 3 pages sur le coût écologique du numérique ne suffisent pas pour rentrer dans le détail et d'autres en parlent déjà bien mieux que nous, nous vous invitons donc à les découvrir en annexe.
Evidemment, le numérique apporte un nombre incroyable de solutions au quotidien, dans le domaine médical, de la santé, de l'éducation, de la culture, du service… Nous aimerions attirer votre attention sur l'envers du décor.

Lorsqu'une vibration fantôme nous fait déverrouiller 150 fois le téléphone par jour… Lorsque vous passez votre dîner entre amis, votre attention est portée sur le smartphone. La peur maladive de rater un événement, relayé par la montre connectée… qui nous déconnecte de l'instant présent. Le problème n'est pas tant le numérique, qui reste un outil, mais plutôt ce qu'on en fait et comment on incite l'utilisateur à s'en servir. Il devient vite facile d'exploiter les fragilités de l'utilisateur, on parle ici du FoMO (fear of missing out) au nom du sacro-saint business model, nous y reviendrons.

Alors comment accompagner les plus jeunes utilisateurs à une meilleure gestion de leur usage du numérique ?

Il est tentant de calmer un enfant en lui mettant dans les mains une vidéo pendant un trajet en métro. La solution est simple et confortable mais lorsque ça devient une habitude, les impacts sont de plus en plus nombreux et dévastateurs : troubles de l'attention, retard du développement du langage, rapport à la réalité et à la relation à l'autre détériorés… La liste est trop longue pour être exhaustive. En tant que professionnels du logiciel, nous avons le devoir de prendre soin de nos utilisateurs (petits comme grands), en restant attentifs aux produits que nous concevons et à l'usage que nous en faisons.

Il est important de cultiver l'esprit critique et de discuter pour se prémunir de ces problèmes avec son entourage, perso comme pro.



Source : Observatoire de l'infobésité et de la collaboration numérique

Nous sommes en permanence inondés d'informations, entre les mails et les réseaux sociaux. D'après Caroline Sauvajol-Rialland, "l'infobésité" est la rencontre entre une personne, son poste de travail et l'organisation (surcharge imposée et interdiction de ne pas répondre).

Couplé aux flots de la vie personnelle (surcharge choisie : on décide la fréquence à laquelle on voit sa grand-mère), le volume de ces stimuli informationnels mène à la confusion mentale, aux risques psycho-sociaux, risques de décision, manque d'innovation (diminution de la créativité)…

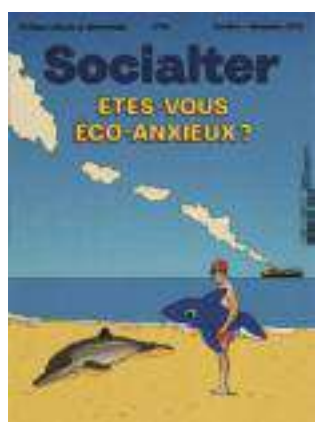
Il s'agit de remettre de la temporalité dans notre pratique info-communicationnelle qui soit éthique, responsable, afin de revenir à des échanges raisonnables et faire valoir le droit à la déconnexion tout en étant exemplaires dans nos sollicitations, c'est-à-dire laisser le temps de répondre sans exiger l'immédiateté.
Via cette infobésité (contraction d'information et obésité), plusieurs problèmes surviennent :

- les fake news (infox / intox)
- l'altération des relations sociales (addiction, cyberharcèlement…).

En tant qu'adultes, nous sommes vulnérables à ces informations toxiques, qu'en est-il de nos enfants ?

Dans une interview, Caroline met en opposition les tâches "liquides", par exemple répondre à un appel, aux tâches solides, le travail de fond permettant l'écriture de cet article. Dans le deuxième cas, nous avons besoin d'une attention accrue pour accomplir le travail, qui devient la valeur intangible que se disputent les GAFAM : on parle d'économie de l'attention.

"Si c'est gratuit, c'est que c'est toi le produit".
On a tous vécu les recherches en ligne qui finissent sur des vidéos de chats en passant par un site e-commerce pour acheter le dernier accessoire indispensable qu'il nous manque… et l'après-midi s'est volatilisée, laissant le goût amer d'avoir perdu un temps précieux on ne sait pas trop où, happé par son écran, scrollant indéfiniment, inconsciemment en quête d'une nouvelle dose de dopamine (hormone du plaisir recherchée par les experts pour activer "le circuit de la récompense", accentuant l'usage excessif des écrans). En effet, les spécialistes travaillent continuellement à nous garder captifs.
Depuis dix ans, les entreprises de la Silicon Valley se sont mises à vendre leurs utilisateurs et leurs données (et non plus des machines et ordinateurs comme c'était le cas avant). Ce sont les annonceurs qui paient pour ces produits, ce sont donc eux les vrais clients et nous qui sommes vendus (car souvenez-vous : si nous ne payons pas le produit, c'est nous le produit). Les annonceurs paient ainsi pour pouvoir avoir accès aux utilisateurs, c'est pour eux le seul moyen de générer des bénéfices.



Si l'on cherche à lister les problèmes liés à la technologie, beaucoup évoqueront le vol de données, les fausses informations, l'addiction, …
Mais le problème est bien plus profond et repose principalement sur le fait que personne ne travaille à diminuer ces conséquences collatérales, puisque cela fait partie intégrante du business model en lui-même.
Tant que le profit restera le critère de performance unique de ces entreprises, les dommages sociaux et environnementaux ne feront pas le poids. **Nous vivons ainsi dans un monde où l'arbre mort et la baleine morte ont plus de valeur que vivants. Incroyable, mais vrai.**

Alors, comme le profit reste l'unique critère de performance, tout est traqué : combien de temps nous restons sur l'image, quelle photos nous regardons, si nous sommes intro ou extravertis, nos traits de personnalité, ce que nous mangeons, aimons, buvons et même les images que nous regardons lorsque nous nous sentons bien ou mal. C'est ce qu'on appelle "le Capitalisme de la surveillance".
Ceci permet ainsi aux algorithmes de prédire notre prochaine action, les émotions auxquelles nous sommes réceptifs ou non, pour qu'ensuite les algorithmes puissent nous soumettre des contenus en alignement avec ce que nous consultons. Cela mène ainsi des milliers de personnes à voter pour un parti plutôt qu'un autre (grâce à ce que l'on appelle "La technologie persuasive" dont le seul et unique but est de changer le comportement d'une personne) ou encore à penser que la terre est plate, ou à mettre notre santé mentale entre les mains de petits émojis dont la GenZ devient dépendante, au point de voir les courbes d'anxiété et de dépression s'envoler entre 2011 et 2013 : + 62% chez les adolescentes les plus âgées, et +189% chez les préadolescents. Quant au taux de suicide, on constate une augmentation de 70% chez les ados de 15 à 19 ans par rapport à la décennie précédente et 150% pour les pré-ados… Tout porte à croire que cela est dû aux réseaux sociaux et aux stratégies de fidélisation qui y sont liées. Les conséquences sont multiples : santé mentale en berne mais également report du passage du permis ou encore rencontres amoureuses plus tardives que les générations précédentes.

La plupart de ces entreprises ont ainsi 3 buts principaux :

1) L'engagement : vous laisser connecté(e) le plus longtemps possible.
2) La croissance : vous devez revenir et inciter nombre de vos amis à utiliser la plateforme.
3) La publicité : générer le maximum de profits pendant que vous utiliserez un de ces services.

Les algorithmes, quant à eux, réfléchissent aux contenus à nous montrer pour augmenter le temps de connexion et le trafic qui y est lié.

En conclusion (le sujet est large et nous vous invitons à consulter les ressources en annexe), nous sommes passés d'un environnement où les inventions étaient de simples outils à un environnement où ils favorisent l'addiction ; où les algorithmes servent leurs propres objectifs et non ceux de l'utilisateur, notamment en faisant appel à des techniques psychologiques, ou encore l'A/B testing dont l'usage principal est parfois détourné de sa raison d'être initiale pour convertir toujours plus d'utilisateurs.

Tant qu'il n'y aura pas de régulation, nous continuerons de légitimer un assouvissement de nos besoins personnels sans nous soucier de notre appartenance et interdépendance au monde, sans prendre notre part de responsabilité (en tant qu'humain, parent, professionnel et citoyen) et de devoir envers le vivant.

Cela affecte l'environnement depuis longtemps mais espérons que la goutte d'eau soit ce qui fasse changer les choses, faisons en sorte que le profit ne soit pas le seul critère de performance. Privilégions ce qui est "juste" plutôt que ce qui est "cool".

RESSOURCES/ REFERENCES :

- ADEME : https://longuevieauxobjets.gouv.fr/reduire-son-impact-numerique?at_medium=sl&at_campaign=Num-Resp_S2_23&at_platform=google&at_variant=
- Convention citoyenne pour le climat : https://propositions.conventioncitoyennepourleclimat.fr/objectif/accompagner-levolution-du-numerique-pour-reduire-ses-impacts-environnementaux/
- GreenIT.fr : les impacts environnementaux du numérique en France: https://www.greenit.fr/2020/06/23/quels-sont-les-impacts-environnementaux-du-numerique-en-france/?
- Rapport de l'étude : l'empreinte environnementale du numérique mondial - GreenIT.fr:https://www.greenit.fr/wp-content/uploads/2019/10/2019-10-GREENIT-etude_EENM-rapport-accessible.VF_.pdf
- Livre blanc numérique et environnement : 26 actions concrètes pour faire converger numérique et écologie - GreenIT.fr: https://www.greenit.fr/2018/03/19/26-actions-concretes-faire-converger-numerique-ecologie/
- Référentiel général d'écoconception de services numériques : https://ecoresponsable.numerique.gouv.fr/publications/referentiel-general-ecoconception/
- Techologie Podcasts : https://imagotv.fr/podcasts/techologie
- Sobriété numérique, les clés pour comprendre et agir : Frédéric Bordage : https://livre.fnac.com/a13445312/Frederic-Bordage-Sobriete-numerique
- Derrière nos écrans de fumée (Netflix)
- Green IT, les clés pour des projets informatiques plus responsables par Margerie GUILLIOT, Raphaël LEMAIRE, Sylvain REVEREAULT
- https://louisderrac.com/2022/08/le-numerique-acceptable/
- manifeste du low tech punk https://www.linkedin.com/posts/tarikbouriachi_manifeste-low-tech-punk-extraits-le-low-activity-7130919266253467648-MsKa/?originalSubdomain=fr

I.2- L'IA est-elle compatible avec la qualité durable ?

par Marc Hage Chahine

1- La qualité durable en quelques mots

1.1- La philosophie

La qualité durable est avant tout, comme l'Agile, un état d'esprit. Elle a pour vocation de répondre à 3 problématiques majeures liées aux services numériques :

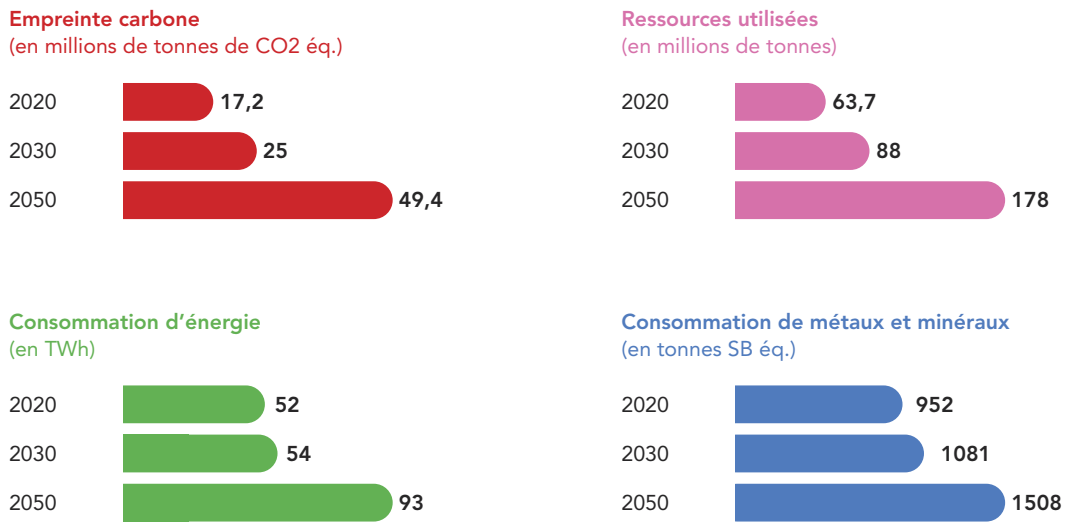
1- les dérives des logiciels vis-à-vis de ses utilisateurs,
2- le désengagement et le mal-être des personnes créant le logiciel,
3- les impacts environnementaux des services numériques.

L'impact écologique grandissant des services numériques

Ce sont notamment les désastres sur la biodiversité liés à l'utilisation des ressources pour construire les terminaux et la consommation de CO2 qui explose.
Quelques chiffres de l'ADEME permettent de se rendre compte de l'ampleur du phénomène qui est en forte croissance :

Des indicateurs issus d'une méthode normalisée d'analyse de cycle de vie qui comporte des définitions précises

Évolution de 4 indicateurs de l'impact environnemental du numérique dans le scénario tendanciel, en valeurs absolues.



Source : https://www.arcep.fr/uploads/tx_gspublication/dossier-presse-Etude-Ademe-Arcep-lot3_mars2023.pdf

Les services numériques se doivent de prendre en compte leur impact écologique car à l'heure actuelle, ils sont plus un problème qu'une potentielle solution qui serait apportée par de « l'IT for Green ».

L'impact des logiciels sur les utilisateurs

Les intérêts des utilisateurs ne sont plus au centre des préoccupations ! Cela engendre de graves dérives pour la santé et leur bien-être.
Cela s'illustre facilement avec l'utilisation et la revente de données personnelles. Ou encore l'avènement de l'économie de l'attention, où le concurrent devient le « sommeil » comme l'a revendiqué le PDG de Netflix… Alors que les problèmes de santé engendrés par un manque de sommeil sont bien connus !
Il est important que les services numériques reviennent à leur but initial : être des services et non des asservisseurs.

Le mal-être / désengagement des équipes travaillant dans le numérique

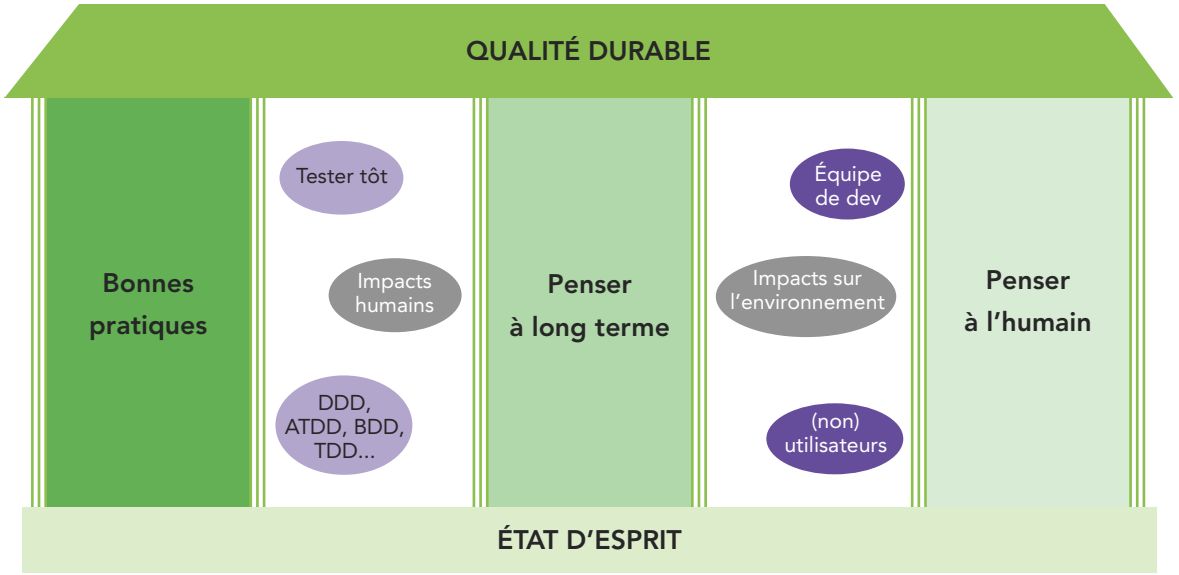
De plus en plus de personnes travaillant dans l'IT travaillent dans un rush permanent. Ce rush les fatigue et les pousse à proposer des services numériques de mauvaise qualité. Cela s'observe facilement avec des dérives de l'Agile où les équipes sont amenées à faire du « quick & dirty ». Tout cela crée un mal-être qui débouche sur un désengagement des personnes, du turn-over ou des burn-out.

La qualité durable souhaite lutter contre ces trois fléaux

Elle vise à proposer des produits pérennes. Par pérennité, il faut entendre : pérenne pour l'environnement, pérenne pour les utilisateurs et pérenne pour les personnes construisant ces services, comme le montre l'image du chapitre d'introduction de cette partie.

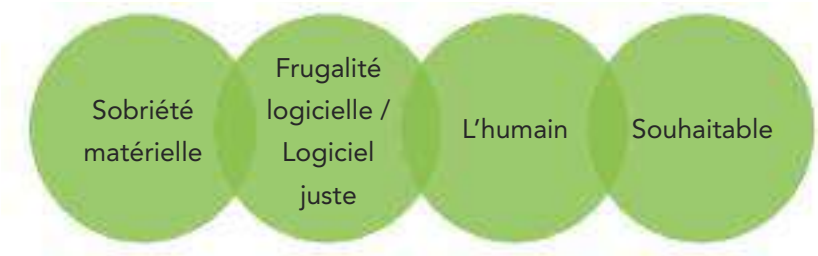
1.2- Les fondations

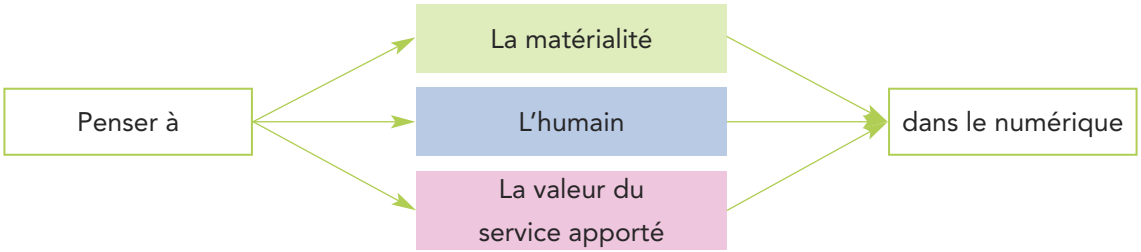
Comme indiqué précédemment, la qualité durable est avant tout un état d'esprit sur lequel se reposent des bonnes pratiques associées à une pensée à long terme et une forte pensée pour l'humain. C'est avec le bon état d'esprit et ces 3 piliers que l'on peut atteindre et maintenir une qualité durable.



1.3- Les valeurs

Comme pour l'Agile, la qualité durable promeut des valeurs qui peuvent être représentées comme ceci :





Afin de proposer des services de qualité durable, le numérique se doit de penser à la matérialité (les éléments physiques utilisés avec ce service), l'humain et bien sûr la valeur que le service apporte.
Afin de répondre à ces nécessités, il est important de penser en termes de coûts réels (financiers, environnementaux et sociétaux) du service, de penser au bien-être des personnes contribuant à créer le service, aux utilisateurs et à l'environnement.

2- L'IA et ses problématiques

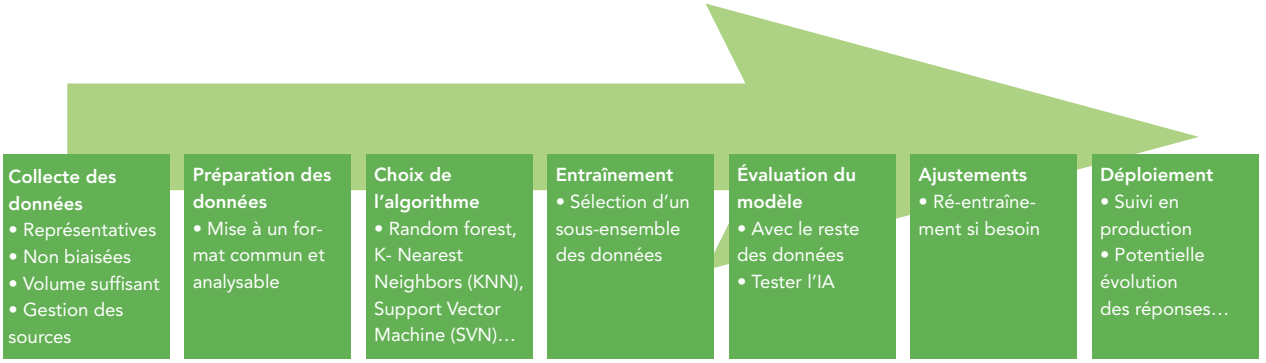
2.1- La complexité de l'IA

L'IA est une technologie très complexe. Quand on parle IA, on pense réseaux de neurones, apprentissage permettant au logiciel d'opter pour des réponses adaptées.
En revanche, on oublie parfois que les IA sont également très diverses. Il y a notamment :

- des IA génératives dont ChatGPT, qui est un LLM (Large Langage Model), est le représentant le plus connu,
- des IA suggestives qui sont là pour faire des propositions à l'utilisateur comme certains algorithmes de suggestion de vidéos,
- des IA d'aide à la décision comme les IA pour prioriser les tests ou de décision judiciaire pour des remises en liberté,
- des IA prédictives pour simuler le comportement futur de divers produits…

De plus, les types d'apprentissage sont également nombreux. On peut penser à des techniques avec ou sans réseau de neurones, avec ou sans supervision…

Enfin, les étapes d'apprentissage par machine learning d'une IA sont nombreuses et complexes :



2.2- Les dérives de l'IA

L'IA n'est pas seulement complexe ! On observe de nombreuses dérives dans son utilisation.

Une des dérives les plus courantes est le fait de considérer que l'IA a forcément raison, donc des utilisateurs qui se fient trop aux résultats de cette dernière.

Cette dérive a engendré de nombreux scandales, comme celui à propos de la remise en liberté de détenus aux USA en 2013, ou l'IA de détection des fraudes aux allocations familiales aux Pays-Bas en 2021.

Malheureusement, ce ne sont pas les seules dérives que l'on peut constater !

Pour entraîner les IA ou même pour les faire fonctionner, de nombreuses données personnelles sont utilisées. De même, les entraînements de certaines IA comme ChatGPT posent question. En effet, l'entraînement de ChatGPT s'est fait à partir de pages web accessibles gratuitement, parfois avec des licences en creative commons interdisant la commercialisation et imposant la mention, parfois en bafouant le droit d'auteur. Ces données servent maintenant à un service payant !

2.3- Les ressources consommées

On peut penser que les 2 premières problématiques peuvent être surmontées avec une montée en compétence et une transparence. Cela est en partie vrai… mais les problématiques de l'IA ne s'arrêtent pas là.

Les IA sont très gourmandes en ressources. L'entraînement de ChatGPT a nécessité plus de 500Gb de données, 10 000 processeurs, la consommation de 700 000 litres d'eau… et évidemment beaucoup d'argent !

2.4- Tester l'IA

Au-delà des problématiques précédentes, il est essentiel de savoir tester une IA. A l'heure actuelle, c'est une compétence rare et totalement insuffisante à la vue des besoins à venir.

Les connaissances théoriques peuvent s'acquérir avec une certification mais il est également impératif d'avoir une expérience pratique.

Enfin, il faut également garder à l'esprit que certaines IA continuent à évoluer en production. Il est nécessaire d'être capable d'anticiper des dérives liées à l'utilisation de ces IA, afin d'éviter des problèmes comme celui connu avec TayTweets :

2.5- L'impact à long terme sur l'industrie du logiciel

Enfin, comme pour l'automatisation, l'IA va permettre d'automatiser des tâches. Les tâches qui sont automatisées par l'IA sont les tâches à faible valeur ajoutée faites par des profils « juniors ». Les profils plus seniors restent obligatoires pour s'assurer du bon fonctionnement de ces IA. Néanmoins, lorsque ces profils seniors partiront à la retraite, il n'y aura plus de relève car les juniors ne seront plus là et n'auront pas pu acquérir d'expérience.

2.6- Conclusion

l'IA ne semble pas compatible avec une qualité durable qui :

- souhaite préserver l'environnement,
- souhaite protéger les utilisateurs des services numériques,
- souhaite protéger les créateurs des services numériques.

Et pourtant…

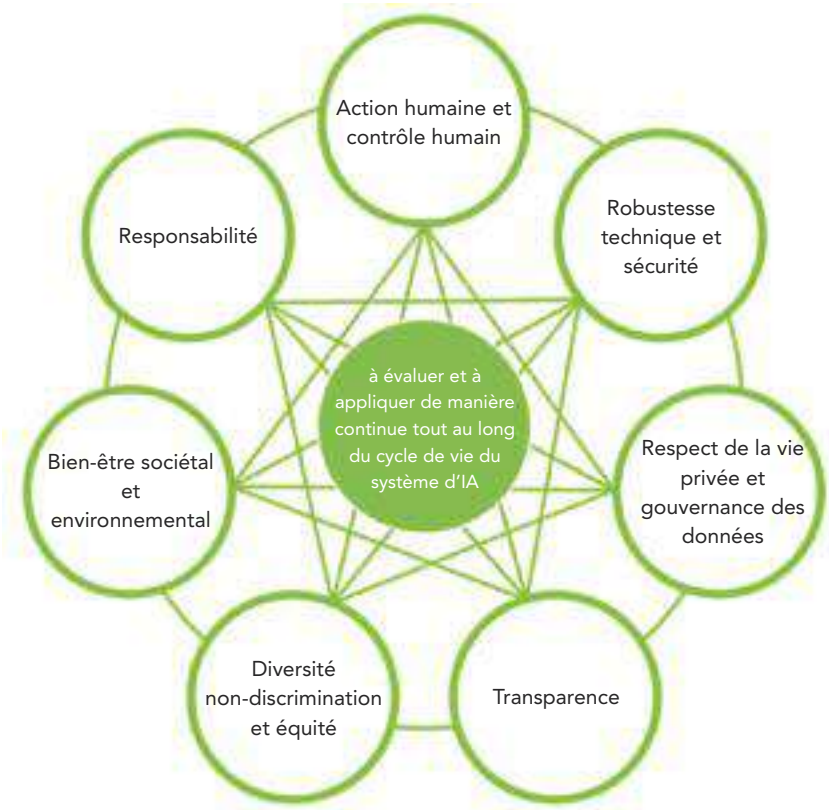
3- Construire une IA compatible avec la qualité durable

Concevoir une IA compatible avec la qualité durable n'est pas une mince affaire. Il existe cependant des éléments ou pratiques qui peuvent permettre de tendre vers cela.

3.1- Construire une « IA de confiance »

L'IA de confiance prônée par l'UE est un élément important si l'on veut répondre en termes d'éthique et d'impact utilisateur aux problématiques de l'IA.

Cette dernière est parfaitement résumée avec ce schéma :



Source : UE guidelines pour une IA de confiance

Une IA de confiance est une IA qui reste sous le contrôle de l'humain, qui est transparente (open source), qui respecte les normes comme la RGPD, l'environnement et les utilisateurs.

3.2- Construire une IA qui suit les principes d'éco-conception (RGESN)

Le RGESN donne des bonnes pratiques et des questions à se poser afin qu'un service numérique soit autant écoresponsable que possible. Même si cela n'est pas lié à la technologie et donc non lié à l'IA, il est essentiel qu'une IA de qualité durable suive ces bonnes pratiques qui sont communes à l'ensemble des services numériques.

Parmi les questions à se poser quand on construit un service numérique écoresponsable :

- Le service numérique a-t-il été évalué favorablement en termes d'utilité en tenant compte de ses impacts environnementaux ?
    - On note l'importance de la valeur environnementale du service. Cette valeur doit être supérieure à son coût.
- Le service numérique utilise-t-il une phase d'apprentissage avec un niveau de complexité minimisé et proportionné à l'usage effectif du service ?
    - Cette question porte sur la proportionnalité des moyens. Il faut utiliser les ressources de manière efficiente !

3.3- Construire une IA qui répond positivement à la méthode BISOU

La méthode BISOU n'est pas liée au numérique mais plus simplement à l'achat d'un produit ou d'un service. Elle s'applique cependant très bien à la construction d'une IA ou même à son adoption.

BISOU sert en fait de moyen mnémotechnique pour avoir une « checklist » et s'assurer du réel intérêt du produit. Les différentes lettres de cette méthode ont pour signification :

- B = Besoin
  - L'utilisateur a-t-il vraiment besoin de cette fonctionnalité / de cet usage ?
- I = Immédiat
  - L'utilisateur en a-t-il besoin actuellement ?
- S = Similaire
  - Existe-t-il un service numérique (potentiellement sans IA) qui répond déjà à ce besoin ? Il faut utiliser l'IA uniquement si on en a besoin pour répondre à une problématique.
- O = Origine
  - D'où proviennent les différents éléments utilisés par le service numérique (hardware et software)?
- U = Utile
  - L'usage est-il vraiment utile ?

Une IA de qualité durable doit répondre à l'ensemble de ces critères.

3.4- Construire une IA qui a un vrai retour sur investissement

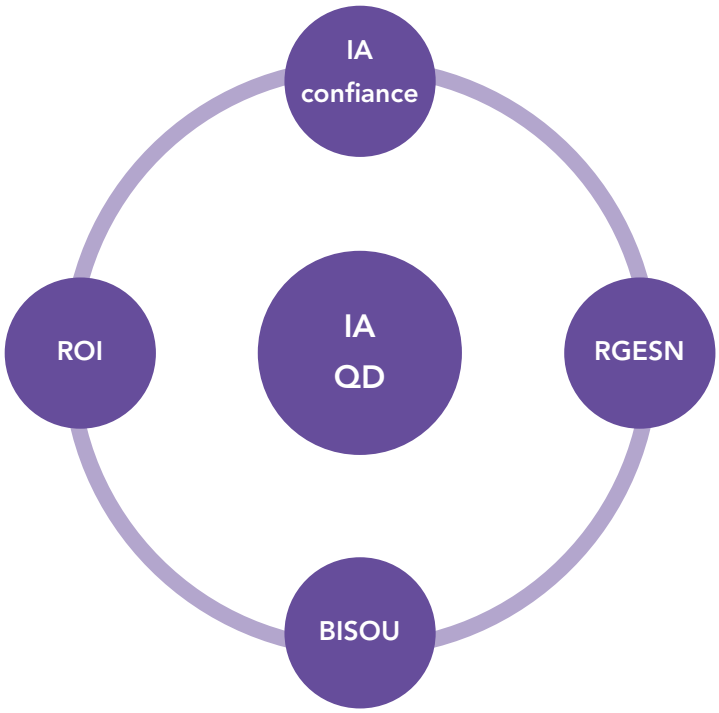
Enfin, construire une IA compatible avec la qualité durable veut aussi dire que cette IA doit être utilisée sur une période assez longue.

Afin de s'en assurer, il est essentiel que cette IA soit rentable et offre un retour sur investissement. Il est important de se rappeler que les investissements sont évidemment financiers mais également humains. En effet, un service entraînant une détérioration des conditions de travail des personnes le créant, et le maintenant, est voué à disparaître. De même, un service non utilisé par les utilisateurs car ils jugent qu'il faut trop de ressources (montée en compétence, temps, argent) pour l'utiliser par rapport à son intérêt, n'a pas d'avenir à long terme.

4- Conclusion

4.1- Une IA compatible avec la qualité durable est possible !

Avoir une IA compatible avec la qualité durable est complexe mais pas impossible.



C'est finalement assez logique : la qualité durable et l'IA ne sont pas en opposition car la qualité durable ne dépend pas de la technologie.

Néanmoins, l'utilisation de l'IA nécessite des précautions et de nombreuses ressources, il est important de bien peser le pour et le contre avant de s'engager dans l'IA… et cela est valable que l'on soit en qualité durable ou non !

Afin de proposer une IA compatible avec la qualité durable, il est nécessaire que cette dernière allie l'IA de confiance (éthique), le RGESN (éco-conception), la méthode BISOU (utilité) et un ROI sur les ressources humaines et financières.

4.2- Les initiatives actuelles pour rendre les IA plus durables

Enfin, comment ne pas finir cet article avec la présentation de certaines initiatives actuelles qui visent à rendre les IA plus durables.

Je pense notamment à :

- L'IA Mistral (Mistral 7B) :
  - IA open source française.
  - IA spécialisée, ce qui permet de limiter les données.
- Ecolab
  - Laboratoire d'innovation au service de la transition écologique du Commissariat Général au Développement Durable.
- Etalab
  - Coordination et promotion de l'action de l'Etat sur l'exploitation des données et des codes sources.
- Ekitia
  - Association de préfiguration d'un pôle d'économie de la donnée.
  - Respect de la RGPD.

Prenons l'exemple de l'IA Mistral 7B.

- Sur la consommation de ressources :

Mistral 7B est une IA génératrice comme ChatGPT. Mais contrairement à ChatGPT qui est un LLM (Large Langage Model), on parle ici de SLM (Short Langage Model). Le principe est de proposer une IA plus simple à entraîner et moins gourmande, en la spécialisant sur un sujet. Cela permet d'avoir de bons résultats sur un sujet spécifique choisi… tout en consommant peu de ressources. Il est possible de faire tourner Mistral sur un simple ordinateur personnel.

Cette économie des ressources rend plus atteignables les objectifs d'avoir un ROI sur le produit proposé ou de répondre favorablement aux critères d'écoconception du RGESN.

- Sur l'éthique :

Mistral 7B est open source ! Cela permet de savoir exactement comme il fonctionne et avec une grande transparence.

- Sur la méthode BISOU :

La spécialisation du produit permet de répondre à un besoin précis (B) sur lequel il n'y a pas d'alternative similaire (S). Le fait que cette IA soit open source et conçue dans l'UE (France) permet également de se rassurer sur son Origine (O).

Pour ce qui est de l'utilité (U) et l'immédiateté du besoin (I), cela dépendra des choix stratégiques faits pour le produit.

I.3- L'écoconception doit devenir la norme pour fabriquer et tester les services numériques d'aujourd'hui et demain

par Guillaume Kerrien

Le Numérique Responsable (ou NR) est-il le grand enjeu de la filière numérique pour la période 2020-2030 ? Ou est-ce un effet de mode, surfant sur le concept de la « RSE » (Responsabilité Sociétale et Environnementale) ? J'aurais tendance à répondre : les deux !

Pour illustrer ma réponse, voici une anecdote, souvenir d'un séminaire d'acteurs publics auquel j'ai eu l'occasion de participer en 2017. Je me rappelle avoir été gêné et avoir eu l'impression de choquer, ou en tout cas de laisser mes interlocuteurs totalement indifférents, quand je leur expliquais que le big data faisait à notre époque ce que l'industrie agroalimentaire avait fait dans les années 1970-1980 : se définir d'abord avec beaucoup trop de données-déchets générées et parfois des « emballages inutiles » (j'ai fait l'analogie entre les emballages alimentaires et les surcouches de données). Depuis, le concept de la sobriété numérique est apparu et il est plus aisé de tenir ce type de discours et surtout d'en débattre. On peut aujourd'hui s'appuyer sur des données d'impact, sur des bonnes pratiques faisant autorité.

Si le sujet est bien plus audible depuis quelques années, afin de le rendre crédible, il est indispensable que le Numérique Responsable s'appuie sur des outils et des méthodes de fabrication et d'évaluation/test. Car c'est bien là l'enjeu de la décennie pour le monde de l'IT, tant les équipements et usages numériques sont en train de prendre une place prépondérante dans notre quotidien, avec un impact croissant sur notre environnement et notre société, pour le meilleur et pour le pire. Dans cet article, je mettrai de côté l'outil indispensable du bon chargé de mission NR : le bilan environnemental et sociétal du numérique, et tous les mécanismes d'évaluation qui vont avec. Je vais surtout évoquer pour vous l'autre grand levier de responsabilisation de nos équipes projets informatiques et métiers : **l'écoconception de services numériques.** Si les cadres de bonnes pratiques se structurent de mieux en mieux, nous verrons qu'il y a encore du chemin à parcourir avant d'aboutir à une norme standard de conception informatique éco et socio-responsable nécessaire.

1 - Le numérique responsable est l'enjeu de cette décennie pour la filière numérique

Pour que le « Green IT » puisse commencer à s'imposer dans le débat des méthodes et bonnes pratiques de conception des services numériques, il a bien fallu attendre une certaine vague de responsabilisation de la filière. Cette approche écoresponsable de l'informatique a déjà plusieurs décennies derrière elle, et les grands concepts restent assez stables depuis les années 2007-2009, où GreenIT.fr et d'autres collectifs pionniers ont commencé à construire des référentiels.

C'est pourtant à l'extérieur du monde informatique que la **prise de conscience collective** s'est le mieux engagée, avec la volonté d'inclure le numérique dans les politiques de développement durable (ou RSE). C'est notamment le fait de voir cette filière comme un problème (contributeur accru dans le bilan carbone des organisations, vecteur de fracture numérique), autant que comme une solution (levier de transition écologique et sociétale). Il y a aussi l'évolution des **données d'impact**, grâce à de nombreuses études souvent portées par l'ADEME (Agence de l'Environnement et de la Maîtrise de l'Energie). C'est ainsi qu'est né le Numérique Responsable, véritable miroir des ODD (Objectifs de Développement Durable) à l'échelle de l'IT, dans l'idée de freiner une certaine course technologique et technique non maîtrisée.

Et alors que les années 2010-2020 ont ainsi fait la part belle au big data, à l'uberisation, grâce aux nouvelles technologies, aux applis mobiles comme au « cloud-computing », les années Covid apparaissent comme un tournant pour la filière numérique. Avec non seulement une explosion des usages (visioconférence, achat en ligne, chat, ...) plus soudaine que prévu et en même temps, une prise de conscience éthique et environnementale plus aiguë. La question se pose de plus en plus de tracer une frontière entre l'utile et le futile, entre l'« IT for good » et l'« IT as usual ».

Il y a désormais urgence à agir et à réorienter le numérique vers une forme de **progrès technologique plus qualitatif et inclusif, plus durable** en somme. La société prend petit à petit conscience des mauvais impacts environnementaux et sociétaux que les nouvelles technologies (IA, IoT, 5G…) nous apportent par effet rebond. Certaines données ou projections sont déjà vertigineuses (*) :

- Le smartphone a révolutionné le quotidien en 10 ans, pour équiper aujourd'hui 9 Français sur 10, avec une inflation du prix moyen de l'ordre de 40 % en 10 ans (et une augmentation du poids moyen de 30%) et une dépendance accrue à ces équipements (on parle de « nomophobie »).
- Le nombre d'utilisateurs d'Internet a plus que doublé en 10 ans dans le monde (plus de 5 milliards en 2022), augmentant avec les besoins en équipements.
- Entre 2020 et 2030, les projections nous annoncent une création de données multipliée par 12 (passant de près de 50 à plus de 600 zettaoctets dans le monde).
- Il faut environ 1 tonne de matière première pour fabriquer l'équipement moyen annuel d'un Français, quand celui-ci produira 300 kg de déchets annuels environ.
- Il est envisagé que la contribution aux GES (Gaz à Effet de Serre) passe de 4 % en 2020 (autant que l'aviation civile) à plus de 7% en 2025, peut-être 14 % d'ici 2040.
- Pourtant, aujourd'hui encore, on estime que 13 millions de Français sont éloignés du numérique, pour ne pas dire purement exclus.

2 – L'écoconception est un levier indirect et puissant de réduction de l'empreinte environnementale

Le besoin d'introduire des notions de sobriété ou de durabilité dans les services numériques s'impose petit à petit comme un levier important. Et pour ce faire, c'est notamment sur la

réduction de la taille (principe d'efficience) et de l'impact (principe de durabilité et sobriété) des services numériques que les concepteurs et développeurs doivent se pencher : écoconcevoir.

C'est malheureusement bien plus complexe que la simple mise en place de nouvelles méthodes de développement (une revue de code Sonar couplée aux bonnes pratiques d'écoconception ne suffira pas). Car le changement est à la fois :

- **culturel** : il faut penser les besoins et les projets différemment, en favorisant la prise de conscience des utilisateurs : vers l'utile, l'essentiel, en cohérence avec des objectifs non contraires au développement durable
- **systémique** : pour que cela fonctionne, il faut une réaction en chaîne, car les services numériques sont rarement indépendants. Au contraire, ils sont de plus en plus interfacés et imbriqués.

Trois approches clés facilitent l'écoconception systémique :

- d'une part, la recherche de la **satisfaction utilisateurs à moindre ressource** : la recherche d'efficience est une démarche qualité plus précise que la performance. Et elle va également plus loin que la simple recherche de la satisfaction utilisateurs,
- d'autre part, un **choix judicieux des technologies et de l'infrastructure** : l'achat responsable à tous les niveaux (matériel, composants technologiques et logiciels),
- enfin et avant tout, une **réorientation des objectifs métiers** selon le prisme de l'utilité. Il s'agit de challenger l'utilité pour les utilisateurs, pour une organisation, pour la société, pour la planète. Je trouve par exemple très pertinent de considérer des personae atypiques en phase de conception : des personnes non/mal voyantes (2 % de la population), des illettrés numériques (15%) et certains projets ont même raison de considérer la planète Terre (ou un territoire), comme un persona à part entière.

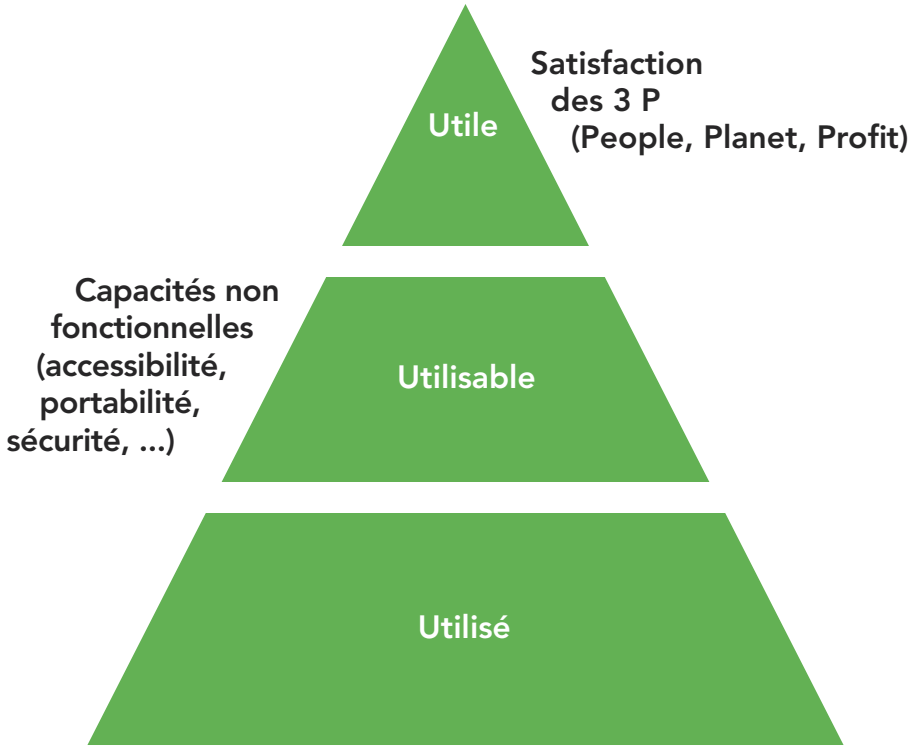
L'architecture et l'urbanisation sont de bons alliés. On pourrait faire le parallèle avec la cloudification des services numériques, qui nécessite également une approche systémique, en repensant l'architecture complète d'un produit et de son écosystème. On ne migre pas son produit en « SaaS » (Software as a Service) sans embarquer tout un ensemble (Infrastructure, Interfaces, Modèle de maintenance, Modèle économique, Sécurité…).

Mais c'est du côté des fonctions métiers, voire des utilisateurs, qu'il faut évidemment aller chercher les meilleurs alliés. Car les services numériques doivent d'abord être des outils utiles, utilisables et utilisés (principe des 3U). Ainsi, pourquoi conserver un service numérique qui ne sera pas ou pas assez utilisé ? Pourquoi concevoir un service qui n'a pas de valeur sociétale ou environnementale ? De ce point de vue, cela peut amener à devoir changer les mentalités des fonctions métiers, adopter de nouvelles habitudes tant chez les utilisateurs que chez les donneurs d'ordre. Voilà pourquoi le Numérique Responsable doit impacter la culture d'une organisation avant même qu'elle envisage de pouvoir écoconcevoir.

C'est tout un système vertueux qui peut s'enclencher avec l'écoconception systémique et orientée vers l'utile : plus sobres et durables seront les logiciels et applications, moins l'obsolescence se fera sentir sur les équipements et infrastructures. Plus responsables seront les usages et les achats, plus sobres et durables seront les nouveaux besoins exprimés. Je suis le premier à soupirer fort (avec le micro coupé) quand un éditeur me présente sa feuille de route d'évolutions de son produit, avec une liste de fonctionnalités longue comme le bras. Seulement, s'il le fait, c'est que ces besoins ont été exprimés, mais il n'a pas auparavant évalué leur efficience et leur impact / utilité réelle, il n'a pensé qu'à la satisfaction (potentielle!) des utilisateurs.

L'enjeu de cette décennie est d'abord celui-ci : réduire le poids du numérique dans nos foyers et dans nos organisations (sachant que l'impact est d'abord à la fabrication/achat, plus qu'à l'utilisation et pour 80 % sur les terminaux, plus que sur l'infrastructure, de quoi avoir bien en tête les ordres de priorités), tout en orientant la filière vers les bons objectifs. Pour cela, il est nécessaire de construire des services numériques durables, d'un point de vue produit, d'un point de vue matériel encore plus, et surtout, d'un point de vue utilité. Avec des effets positifs prévisibles pour l'inclusion : si le matériel dure, si le besoin perdure, meilleures seront l'adoption et l'adaptation des utilisateurs dans le temps.

L'écoconception est finalement un levier d'action indirect, dans le sens où il doit avant tout amener l'utilisateur final à acheter mieux, à faire durer son matériel et à s'orienter vers l'essentiel :



(*) Sources : Ademe-ARCEP, Statista, CREDOC, GreenIT.fr

3 – Comment les méthodes d'écoconception se structurent-elles ?

Comme toutes les approches nouvelles, pour bien commencer, il vaut mieux d'abord se former et s'appuyer sur l'état de l'art. Et en matière d'écoconception et de Numérique Responsable, il y a eu une véritable révolution intellectuelle ces dernières années. A tel point que désormais, l'on peut presque s'y perdre, tant les outils et référentiels se sont développés. Fort heureusement, les experts du NR comme les pouvoirs publics les plus concernés l'ont bien compris et travaillent à la bonne structuration de cette nouvelle discipline.

Du point de vue du cadre légal sur le volet environnemental du Numérique Responsable, les lois REEN (pour une Réduction de l'Empreinte Environnementale du Numérique) et AGEC (pour l'Anti-Gaspillage et l'Economie Circulaire) permettent aux organisations de définir un cadre de fonctionnement, allant plus loin que les attentes RSE (cadrées par la norme ISO 26001).

Concernant le cadre méthodologique de l'Ecoconception, c'est à présent l'ARCEP et l'Ademe qui ont pris en main un des référentiels de bonnes pratiques qui commence à faire autorité : le RGESN (Référentiel Général d'Ecoconception des Services Numériques). Et ce dans la continuité du travail collaboratif qu'avait initié la DINUM (Direction Interministérielle du Numérique), mais aussi en s'appuyant sur des cadre plus anciens, tels que les bonnes pratiques de GreenIT.fr ou le référentiel de l'INR (Institut du Numérique Responsable), une association qui regroupe les organisations les plus actives sur le terrain du Numérique Responsable. Le grand avantage de ce RGESN est qu'il s'inscrit aussi bien dans des thématiques techniques que sociales ou éthiques, en cohérence avec le respect des règles d'accessibilité, de sécurité, d'interopérabilité ou encore de protection des données personnelles.

Le RGESN est pensé pour couvrir l'ensemble du cycle de vie d'un projet de conception de services numériques, que ce soit pour des sites Web, pour des développements internes à une organisation ou pour des solutions d'éditeurs avec ou sans technologie innovante (IA ou IoT par exemple). Il y a ainsi 91 critères à étudier, pour 9 familles d'actions (de la stratégie au développement, en passant par l'hébergement ou la définition des contenus). Dans le déroulé d'un projet, les différentes lignes du RGESN (qui s'apparente à une matrice de critères) permettent d'alimenter les phases et les livrables associés. Par exemple, certaines lignes vont aider à décider des lignes conductrices du dossier de cadrage, d'autres lignes vont alimenter une matrice d'exigences, entrant du cahier des charges ou des spécifications du service numérique. Il peut paraître regrettable de ne pas y trouver une famille du test, mais il y a quelques critères en lien. Et finalement, un spécialiste du test pourra trouver dans ce référentiel un formidable entrant pour construire une stratégie de test et alimenter ses exigences projet et produit !

Toute la difficulté est que ce cadre générique peut apparaître comme trop abstrait pour certains acteurs projets, avec un vocabulaire pas assez adapté au cadre méthodologique déjà en place et ancré dans l'organisation. Il est donc important de diffuser ce référentiel en tenant compte de la réalité du terrain, des projets, quitte à adapter le contenu. Une simplification voire une reformulation des critères peut avoir du sens, pour ne retenir que l'essentiel. La répartition de ce référentiel au sein de différentes équipes peut également être très utile. J'ai par exemple eu

l'occasion de proposer aux chefs de projet de mon organisation un extrait des exigences clés pour contractualiser du logiciel en « SaaS ». J'ai ensuite réalisé une checklist de 10 questions clés issues du RGESN, que tout porteur de projet numérique doit se poser lors de la phase de cadrage.

Monter en maturité sur la compréhension et l'application de ce référentiel est très important. Il doit permettre de fixer des objectifs atteignables et calés avec la réalité des méthodes de fabrication ou d'achat de logiciel en place, pour accompagner la transformation vers l'écoconception systémique. C'est d'ailleurs la raison pour laquelle un des critères de ce RGESN est le suivant : « Le service numérique a-t-il au moins un référent identifié en écoconception numérique ? ». Quand je questionne des éditeurs, c'est bien souvent la première question que je pose !

Le RGESN n'est pas le seul outil en place. On peut par exemple citer le guide d'écoconception des Designers Éthiques et bien sûr le référentiel pionnier : les 115 bonnes pratiques d'écoconception Web de GreenIT.fr. Ce premier reste cependant une promesse de futur standard qui pourrait permettre une transformation d'ampleur de nos organisations et des acteurs clés du numérique (éditeurs, ESN, services publics numérisés, ...). Une suite logique aux RG* (RGPD, RGAA, RGS...).

4 - L'écoconception a besoin d'un cadre normatif

Si le RGESN propose un cadre très utile aux projets numériques, avec un ensemble de bonnes pratiques pour se poser les bonnes questions, ce n'est pas (encore?) un cadre normatif. Quelle sera la suite ? Va-t-on aboutir à une norme ISO de l'écoconception de services numériques ? C'est l'horizon qui semble se profiler, avec un contenu plus générique, à travers les travaux de l'Afnor, rassemblant une grande partie de l'écosystème Numérique Responsable.

Actuellement, on peut malgré tout s'appuyer sur plusieurs normes ISO afin de normaliser ses méthodes de fabrications. Cela peut devenir nécessaire à l'échelle d'une grande organisation, ou pour justifier la qualité du processus de fabrication d'un produit logiciel. Citons quelques standards intéressants à prendre en compte :

- le **standard ISO 14044:2006** cadre les exigences et principes clés d'une analyse de cycle de vie, cela pouvant s'appliquer à la définition d'un service numérique. L'ACV prévoit des étapes de définition des objectifs/unités fonctionnelles du produit, d'inventaire du cycle de vie du produit/service, puis d'évaluation et analyse,

- Le **label Numérique Responsable** (aujourd'hui attribué par l'association Lucie) est un standard récent qui s'appuie sur la norme "RSE" (**ISO 26001**) pour accompagner et auditer la structuration d'une organisation vers une démarche Numérique Responsable. Ce label est donc plus orienté vers la qualité de l'organisation,

- La **norme ISO 50001** cadre le management de l'énergie dans le but de mieux gérer la consommation énergétique et les dépenses, tout en contribuant à la réduction de l'empreinte écologique d'une organisation. Elle permet plus particulièrement d'accompagner les hébergeurs vers une meilleure performance énergétique de leurs data centres,

- Le **standard ISO 20400:2017** dresse les lignes directrices de l'achat responsable, en lien avec l'ISO 26001,
- Sans oublier la **norme ISO 27001** qui cadre les exigences attendues en matière de systèmes de management de la sécurité de l'information. Cela permet de cadrer la capacité de l'organisation à se protéger des cyberattaques et à se rétablir en cas d'incident (résilience).

Et que dire de la **norme ISO/IEC 25010** cadrant la qualité des produits logiciels ? Si l'on regarde les composantes du modèle de qualité produit, il est regrettable de ne pas y retrouver des caractéristiques telles que la sobriété ou la frugalité. Relevons malgré tout certaines catégories qui sont en lien avec l'écoconception :

- accessibilité (à toute forme d'utilisateurs),
- sécurité (protection des données, notamment personnelles),
- portabilité (sur différents terminaux, dans une approche modulaire).

Dans une version plus ancienne de cette norme, une caractéristique intéressante était positionnée sur chacune des familles de qualité : la conformité réglementaire. Il pourrait être intéressant de considérer de cette façon les valeurs de qualité liées à l'écoconception. Une forme de « conformité environnementale et sociétale ». Car si l'on regarde, par exemple, le RGESN, on se rend compte qu'il permet de répondre à des problématiques de qualité larges.

Une autre piste d'évolution des standards de méthodes de test vers la qualité durable est la prise en compte de la notion de « qualité d'usage » (quality in use). Il s'agit d'un modèle de définition de la qualité qui reste encore assez peu connu et pourtant inclus dans la norme ISO 25010. Si l'on regarde ses composantes, on se rend compte que celles-ci peuvent permettre de cadrer une stratégie de test orientée vers l'utilité et la satisfaction utilisateur en même temps.

On retrouve notamment les caractéristiques suivantes :

- L'**efficacité de l'usage**, en considérant l'ensemble du cycle de vie et des parties prenantes.
- L'**efficience**, en considérant l'atteinte des objectifs en fonction des ressources mobilisées, de la production à l'utilisation.
- La satisfaction, incluant l'**utilité**, la **confiance**, le **plaisir** et le **confort**. On peut ici faire le parallèle avec les 3U. Il faut d'abord chercher à être utile. Il faut aussi que le service numérique soit utilisable. C'est par exemple la sécurisation du service qui peut garantir la confiance des utilisateurs. La portabilité et l'interopérabilité peuvent apporter du confort. Enfin, un service compréhensible et attractif peut apporter plus de plaisir à l'utilisation.
- L'**atténuation des risques**, notamment économiques, environnementaux ou associés à la santé ou la sécurité. La considération de la planète Terre (ou du territoire) comme une partie prenante à part entière est, par exemple, une façon de considérer le besoin du point de vue de l'impact sur celle-ci. On va alors penser le service numérique du point de vue de cet impact et des effets associés (prévisibles).
- La **couverture du contexte d'usage**, avec une logique de flexibilité.

J'ai découvert ce modèle de la qualité d'usage sur le tard (malgré des années d'expertise en qualité logicielle) et je trouve sa définition inspirante. Elle permet de rechercher les caractéristiques de la qualité d'un produit écoconçu. Tout l'intérêt étant qu'on peut l'utiliser pour mettre en place des pratiques de tests très nouvelles :

- la mesure de l'empreinte environnementale sur l'ensemble du cycle de vie du service numérique (dans une logique de recherche d'un « ROI » environnemental qui validerait l'efficacité et l'efficience du produit,
- l'évaluation de la pertinence du service, suivant des caractéristiques d'utilité, de confiance (sûreté, éthique) ou même de plaisir apporté,
- l'évaluation de la couverture des risques environnementaux ou sociétaux,
- la recherche du low tech en visant un niveau d'efficience chiffré (par exemple diviser l'utilisation des ressources par 4 par rapport au fonctionnement existant/attendu du service).

On se rend bien compte de la puissance du cadre normatif comme levier de développement de l'écoconception. Si l'on part sur un cadre de valeurs et de fonctionnement commun pour la fabrication et le test des services numériques, il devient plus aisé de déployer et évaluer la qualité du processus d'écoconception, la bonne couverture des exigences du RGESN, par exemple. Il devient donc important d'aller vers un standard qui pourrait faire autorité en matière de qualité durable. Cela pourrait même permettre d'introduire des indicateurs de durabilité des services numériques.

5 – Comment tester et valider un service numérique supposé écoconçu ?

Si l'on parvient à définir plus précisément le quoi, ce qu'est la qualité d'un service numérique écoconçu, il devient plus aisé de penser le comment : la stratégie de test. Mais il est difficile de répondre à cette question de façon générale, tant les tests attendus peuvent dépendre du type de service numérique et de ses objectifs.

Les conseils et les orientations que je peux relever sont à ce jour les suivants :

- S'appuyer sur des données d'évaluation environnementale et sociétale :
  - Petit à petit, notamment grâce aux études soutenues par l'Ademe, des données d'impact sont disponibles, à ce jour regroupées dans ce que l'on appelle la base Empreinte.
  - Mais il existe d'autres référentiels de données d'impact, comme par exemple celui que le collectif Boavizta maintient en s'appuyant notamment sur les données des fabricants.
  - Certains acteurs privés qui ont développé leurs outils de mesure d'impact ont parfois leur propres modèles, les sources restant globalement autour de l'Ademe, du projet de recherche NegaOctet ou du collectif Boavizta.

- Attention au changement de référentiel de données et au mélange des genres ! Chaque outil a ses sources, chaque source a sa méthode, on peut ainsi obtenir des évaluations différentes pour un même service. L'enjeu est de maîtriser ses moyens de mesure et de rester sur une méthodologie stable, ce qui permettra d'aller dans le bon sens : la recherche avant tout du constat d'une réduction de l'impact.

- Privilégier les phases très en amont et très avales pour tester l'écoconception :

  - Vérifier la couverture des exigences RGESN doit se faire dès le départ, à la conception. La revue de conception est donc un moyen pertinent de s'assurer que les attentes d'écoconception sont bien prises en compte dans la spécification. On peut aussi faire appel à des méthodes éprouvées en matière de design, telles que l'UX ou même le Design de service.

  - Et si l'on veut tester l'efficacité et l'efficience du service, des tests de performance ou une évaluation des ressources utilisées ne peut avoir de sens que si les utilisateurs font partie du contexte de test. Tout du moins, il faut être au plus près des conditions opérationnelles dans un temps de mesure suffisamment conséquent. Car la satisfaction utilisateur peut se dégrader dans le temps et des comportements déviants peuvent aussi apparaître et remettre en cause les hypothèses de ressources mobilisées. On parle souvent d'effet rebond et la réalité de l'usage ne peut se mesurer effectivement qu'après l'apparition de ce phénomène. Dès lors, le principe du bilan d'usage régulier peut être un excellent outil d'évaluation pour permettre l'ajustement du service numérique, cela supposant des tests en prod (sacrilège ?!), et l'usage potentiel de panels d'utilisateurs, via mesures ou enquêtes (en veillant à une représentativité optimale et inclusive/diversifiée).

- Considérer un service numérique dans son ensemble :

  - Sur la totalité de son cycle de vie : depuis les matières premières, composants, matériels mobilisés jusqu'à la fin de vie du service et de ses composants, en passant par les grandes étapes de la fabrication, la distribution et l'utilisation.

  - Sur un ensemble d'utilisateurs cibles représentatif : notamment grâce à une grande diversité des personae et si possible des panels de testeurs.

  - Enfin, sur un ensemble d'objectifs à atteindre, autant que possible en lien avec les objectifs de développement durable. Il n'est donc pas uniquement question de la satisfaction utilisateur, il s'agit davantage d'atteindre un objectif et de viser un progrès qualitatif.

Conclusion

L'écoconception est une révolution qui s'impose. Il faut aller vite pour permettre une adoption massive de ce concept dans nos organisations. Mais cela suppose de s'entendre sur des standards de fabrication et de validation, tout autant que cela nécessite une transformation technique et culturelle au sein de nos organisations déjà bien complexes.

En effet, il faut aller vite. Car deux nouvelles vagues sont déjà en train d'arriver et devraient totalement bouleverser notre vision de la qualité des services numériques : l'Intelligence Artificielle est en train de rentrer petit à petit dans notre quotidien, comme le Smartphone l'a fait il y a une dizaine d'années. Sauf que cela devrait totalement changer la donne en matière d'écoconception. Les premières études laissent à penser que l'impact est totalement différent, peut-être avec une inversion des ordres de grandeur d'impact entre la fabrication et l'utilisation (cela revenant à privilégier la frugalité dans les valeurs d'écoconception).

Et si l'on en croit les projections de l'Ademe et de l'ARCEP, les objets connectés (IoT) devraient également s'imposer et envahir notre quotidien dans les décennies à venir (le tout-digital sera-t-il remplacé par le tout-vocal ?). Ce pourrait être également un changement total de paradigme, avec pour le coup un lien très fort entre l'écoconception et les choix de matériels. Mais là également, on peut s'attendre à une explosion des volumes de données traitées.

Dans tous les cas, le principe de la qualité durable et éthique peut aider à ce que ces changements technologiques profonds aillent dans le bon sens : l'utilité, la frugalité et la sobriété avant tout, la maîtrise des risques et la recherche d'efficience également.

I.4- Collaboration et outils visuels

par Elodie Bernard et Benjamin Butel

Nous allons vous faire découvrir de la manière la plus réaliste possible le quotidien d'une équipe Agile, à travers le regard de Marie, une testeuse de l'équipe. Elle partage durant une semaine son quotidien personnel et professionnel. La narration enchaîne ainsi des bribes de sa vie dans un environnement de travail hybride intégrant du télétravail. Le tout mettant en valeur des méthodes projets basées sur l'usage de visuels.

Par ce magnifique lundi matin d'hiver, Marie déjeune avec son fils Charlie et ils jouent à lister les Pokémons de la première génération.
"Dis Maman, tu vas faire quoi aujourd'hui ?" demande Charlie de sa douce voix.
"Je vais retrouver mes collègues pour travailler en visio. On discutera de ce que l'on fera dans la semaine. Allez, oust petit garnement ! Il est l'heure d'aller à l'école."

Marie est testeuse chez un éditeur logiciel au sein d'une équipe de cinq personnes.
Après s'être servi un café, elle consulte ses réseaux sociaux préférés en quête d'un article intéressant sur le test puis elle attend l'arrivée de ses collègues sur leur outil de visio. Comme souvent, Gérard, développeur frontend, arrive en second, puis c'est au tour de Paula, la Product Owner, Lily la développeuse backend et Pascal, le couteau suisse de l'équipe. Marie admire beaucoup Pascal en raison de ses nombreuses compétences en développement et OPS, son calme, sa gentillesse, sa sérénité et surtout sa pédagogie. Elle aime ce moment où ils se retrouvent le lundi matin pour échanger sur ce qu'ils ont fait le weekend. Ça lui procure un sentiment de bien-être.

Ce matin-là, Paula leur présente le nouveau projet de leur client concernant la mise en place d'un outil de gestion pour les remboursements des frais de déplacements.

PROBLÈME / BESOIN

Légende

Pourquoi
Quoi
Qui
Comment
Objectif

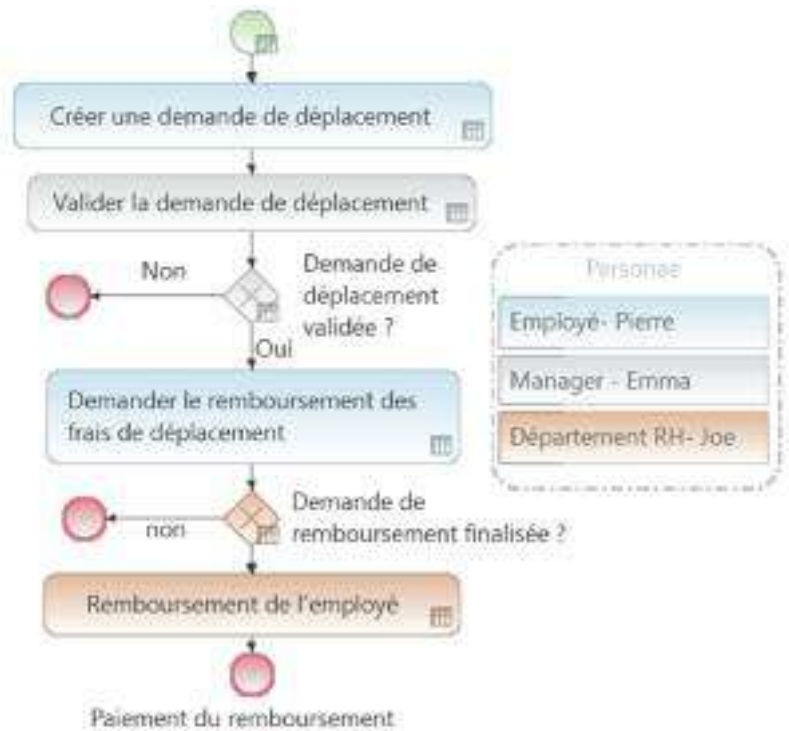
Afin de gagner du temps lors des traitements des remboursements de frais de déplacement, le produit va résoudre la déclaration et le remboursement des frais de déplacement que rencontrent les RH chez notre client en leur donnant un outil clé en main, ce sera un succès si les RH et l'ensemble des utilisateurs arrêtent d'utiliser Excel comme c'est le cas aujourd'hui.

Présentation du nouveau projet sous forme visuelle

Paula aimerait que ce sujet démarre dès cette semaine. Après avoir challengé le backlog avec les autres sujets, l'équipe décide de faire le kickoff dans l'après-midi et l'atelier de découpage fonctionnel le lendemain matin. Elle planifie aussi un example mapping le mercredi et un atelier de découpage technique le jeudi suivi d'une réflexion sur la testabilité.

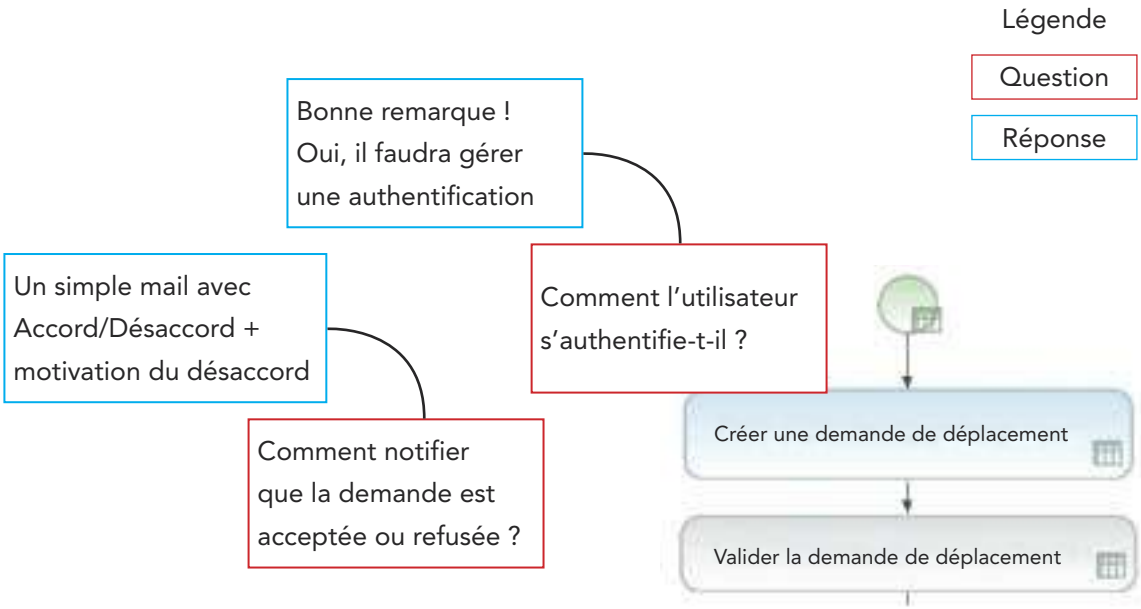
"Paula, tu serais disponible ce matin pour qu'on prépare le kickoff ensemble stp ?" demande Marie.
"Avec plaisir. Je me coule un p'tit café et on peut démarrer!".
Elles se retrouvent en visioconférence avec leur tableau numérique. Elles posent ensemble le cadre du projet et Marie propose d'utiliser Yest pour représenter "grosse maille" le parcours utilisateur.

Nos deux expertes échangent pendant une heure afin d'obtenir une proposition qui leur semble pas mal du tout. Elles se mettent d'accord sur l'animation de ce kickoff. Paula expliquera le contexte et Marie le parcours utilisateur.



Parcours utilisateur avec les personae

Une fois l'équipe au complet, Paula présente le contexte. Marie enchaîne en expliquant le parcours utilisateur envisagé.
Comme d'habitude, l'équipe pose ses questions au fil de l'eau en les ajoutant en rouge. Puis les parcourt toutes pour y répondre, en bleu.
En une heure, l'équipe a pu s'approprier le contenu et chacun va prendre le temps de digérer cette information. Rendez-vous demain matin pour parler du découpage fonctionnel.



Extrait du Kick-off
En rouge les questions, en bleu les réponses

Ce mardi matin pluvieux, l'équipe se retrouve sur site comme prévu et commence la journée par le découpage fonctionnel. Pour l'occasion, Paula avait réservé la salle avec le Meeting Board Klaxoon. Marie aime cette salle dont l'ambiance est cosy avec le canapé et qui permet d'être à la fois en mode décontraction mais invite aussi chacun à participer en se levant et en interagissant directement sur le Meeting Board.

"Bonjour tout le monde. La forme ?", dit Paula puis elle poursuit: "Faisons comme d'habitude !".

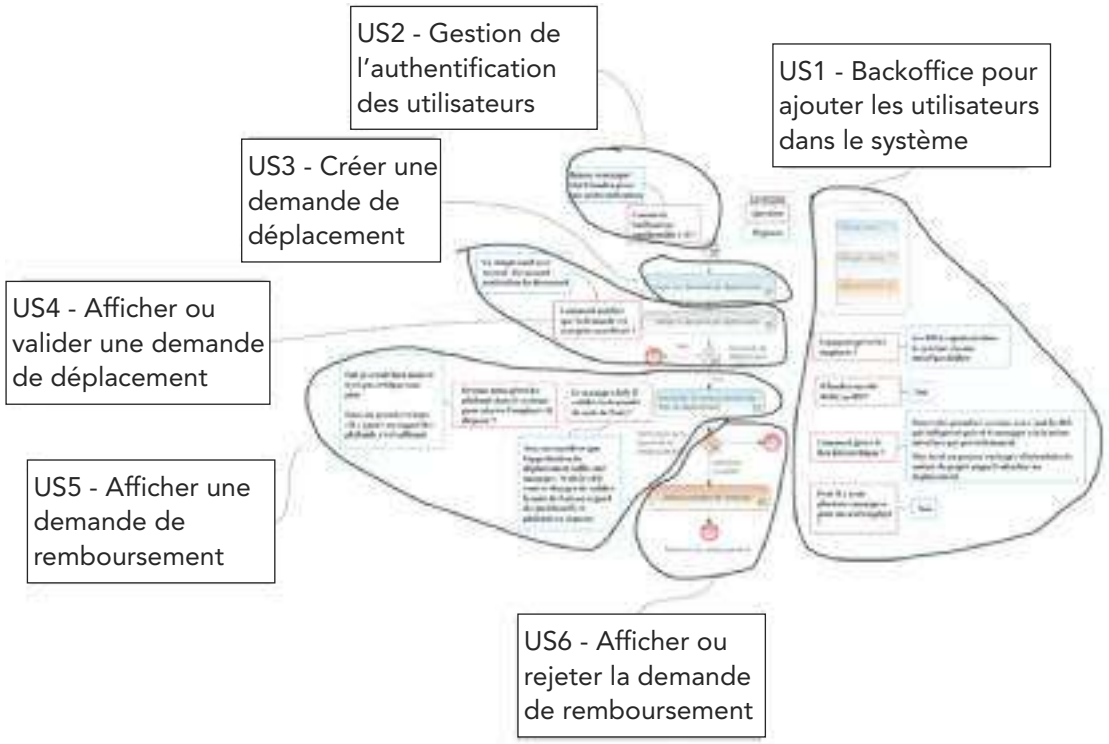
A tour de rôle, chacun identifie un ensemble fonctionnel puis échange sur la nécessité de garder en l'état ou de redécouper. Pascal fait la première proposition : "Pour moi, y'a tout un ensemble lié aux utilisateurs avec une sorte de back office. Je le note US1". Tout le monde est en phase avec ça.
Marie enchaîne en proposant une US2 concernant l'authentification des utilisateurs.

Lily propose ensuite une US pour gérer les demandes de déplacement. Et ainsi de suite, en découpant en deux les US complexes, par exemple, une US pour créer une nouvelle demande déplacement et au total, l'équipe se projette sur 6 US. L'expérience de Pascal l'incite à proposer un ordonnancement :

"Commençons par le cœur du système avec l'US5 suivi de l'US6 ! Ça permettra de valider rapidement avec le client qu'on n'a rien omis dans la gestion d'un remboursement de frais de déplacement.

Ensuite, j'hésite entre l'US2 sur l'authentification des utilisateurs et les US3 et 4 sur la gestion des déplacements. Paula, d'après ton échange avec le client, son besoin premier doit-il absolument embarquer la gestion des déplacements ?".

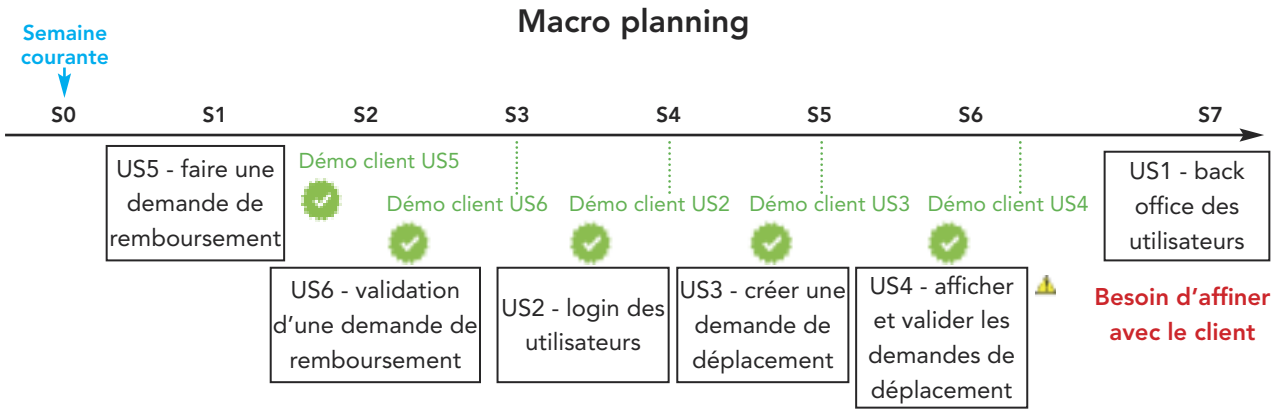
“Pour moi, le besoin premier est de dématérialiser le remboursement. Donc pouvoir rattacher le remboursement à un déplacement validé peut se faire dans un second temps.”, précise Paula. “OK”, dit Marie. “Donc on peut partir sur les US5 et 6, puis l’US2, puis l’US3 et 4 et on termine par le back office avec l’US1. On est tous d’accord la dessus ?“. “Oui”, répondent-ils tous à l’unisson.



Résultat de l’atelier de découpage fonctionnel
Chaque US regroupe des éléments de parcours et les questions/réponses du kick-off

Lily enchaîne et propose qu’on se projette sur une première démo en fin de semaine suivante, par exemple, le vendredi. L’équipe est en phase avec cette proposition et Paula note tout cela.

“Je vous propose de matérialiser ce macro planning et échéance de démo dans notre Board de suivi. Et du coup, on peut faire l’example mapping demain sur l’US5. Comme ça, je peux le préparer dès cet aprem. Si quelqu’un veut le préparer avec moi ?” Lily se propose d’aider Marie. Il est 11h15, l’équipe prend quelques minutes pour parler de tout et de rien autour d’un café.



Macro planning pour le projet “Gestion des notes de frais”
Chaque itération d’une semaine est représentée avec les US à développer

	Equipe	Projet “Gestion des notes de frais”	Projet “Gestion de ses courses”
Lundi	Planning	Kickoff 	Prise en compte des retours de la revue de code 
Mardi		Atelier de découpage fonctionnel 	 Fixer les tests auto
Mercredi		US5 Example mapping US5 Stratégie de test	Plan de test Mise à jour des cas de test
Jeudi		Architecture générale US5 conception des tests	Fixer les tests auto
Vendredi		US5 dev US5 conception des tests	Fixer les tests auto
Lundi	Planning	US5 dev US5 conception des tests	Préparation de la démo
Mardi		US5 dev US6 Example mapping	 **Démo client à 14h**

Exemple de planning

Représentation des différents projets en cours avec les activités critiques à mener

En vert, les activités déroulées comme prévu et en orange les activités en alerte

Marie s'installe devant son bureau et ce matin, c’est *example mapping*. La réunion commençant à 10h30, elle en profite avant pour planifier sa campagne de test pour un autre projet : elle a l'habitude, elle sélectionne ses tests, les ordonne pour l'exécution. Elle se rappelle d'ailleurs qu'elle devait mettre à jour les tests. 10h25, "Déjà! J'ai l'impression de n'avoir rien fait .... Bon, je vais quand même me préparer un thé avant l'*exemple mapping*, ça me fera du bien", se dit-elle !

Tout le monde est là, l'*example mapping* commence par l’US5 comme proposé par Pascal. Paula a préparé avec Marie les règles à présenter. Paula démarre la présentation quand son téléphone sonne.

"C'est le client pour le projet Gestion de ses courses, on n’arrive jamais à se capter, il faut vraiment que je prenne l'appel, j'ai plein de questions pour lui, désolée je vous abandonne, vous faites comme d'hab pour les questions/réponses, Marie tu gères ?".

Après avoir acquiescé, Marie poursuit la présentation. L’équipe a pour habitude d'écrire ses questions en rouge sur les exemples ou les règles de gestion. Puis Paula donne les réponses en vert. Au bout d'une heure, il y a déjà pas mal de questions : Paula va avoir du travail. En attendant,

tout le monde peut retourner à ses activités, il y a déjà assez de matière pour commencer le découpage technique, définir une macro stratégie ainsi que concevoir les tests.

12h30, Nathan, le compagnon de Marie passe sa tête dans l'entrebâillement de la porte pour la prévenir que le déjeuner est prêt.

Après ce repas où Marie a planifié son weekend avec Nathan, elle initie la stratégie de test en utilisant un template lui garantissant qu'elle ne va rien oublier : les tests fonctionnels, de sécurité, de performance et compatibilité de navigateurs. Le client a bien insisté sur la compatibilité navigateurs. Marie prépare la liste de tous les navigateurs à supporter pour s'assurer que les tests seront bien exécutés sur chacun d'eux. Et elle complète aussi la partie "tests fonctionnels" avec uniquement les règles métier à vérifier. Maintenant, il ne reste plus qu'à demander aux autres de relire et compléter. Elle demande à Pascal ce qu'il serait pertinent de faire pour les tests de sécurité et performance sur lesquels elle est moins à l'aise.

Durant 30 minutes, Marie et Pascal discutent des tests de performance et définissent les cas à tester. Lily les rejoint afin de les faire profiter de son expertise sur la sécurité. À eux 3, ils complètent le tableau et sont prêts pour les tests. Marie présente la stratégie à Paula afin de garantir qu'il n'y a pas d'oubli majeur. Entre temps, Paula a apporté les réponses aux questions posées durant l'*example mapping*. L'équipe a désormais une bonne vision de cette *US5*.

8h32 : C'est la course, ce matin !

Nathan est à Paris et habituellement, c'est lui qui dépose Charlie à l'école le jeudi. Marie arrive donc dans l'espace de coworking un peu plus tard qu'elle l'avait prévu, mais ce n'est pas grave, elle a le temps de regarder l'*example mapping* que Paula a complété avant leur point de 10h.

10h arrive, Marie et Paula se posent sur les poufs ultras moelleux de la salle de réunion et Marie affiche le Board Klaxoon sur l'écran pour discuter de la stratégie de test avec Paula.

Paula remarque qu'il n'est pas prévu de test d'usabilité. Et elle se rappelle qu'elle avait discuté avec le client du fait qu'il employait des personnes malvoyantes et que l'interface du site était extrêmement importante. En quelques clics, Marie ajoute une case à son Board sur les tests d'usabilité et Paula complète le backlog avec une nouvelle tâche pour s'assurer de bien intégrer la compatibilité avec un *screen reader*.

Finalement la matinée est presque terminée et Marie et Paula n'ont pas eu le temps de discuter des tests fonctionnels pour tester les premières *US* développées. Mais il est presque 12h et Paula n'a pas particulièrement envie de déborder sur la pause déjeuner. Elle propose donc à Marie de consulter le parcours qu'elle a fait pour l'*US5* et de revenir vers elle si elle a des questions. Marie acquiesce et Paula lance le débat de la journée : "Tu veux qu'on mange où à midi ?"

Extrait d'une stratégie de test en construction
Chaque bloc regroupe le type de test à réaliser

En ce début d'après- midi, Marie retrousse ses manches et attaque la conception des tests pour le nouveau projet ! Elle aborde cette activité avec un enthousiasme sincère car le testing de manière générale et la conception des tests a beaucoup évolué ces dernières années.

Elle est loin l'époque où elle devait parcourir des specs interminables avec des pseudos légendes de couleurs et annotations incompréhensibles et réussir à partir de ça à écrire des tests dans son tableur Excel. Quelle galère ! Là, elle peut se baser sur le brainstorming pour avoir du contexte, sur l'*exemple mapping* pour voir les différents exemples et sur le parcours que Paula a fait pour décrire l'*US5*.

Marie a donc toutes les clés pour commencer sa rédaction de test. Elle récupère directement le parcours de Paula sur Yest et elle va l'enrichir pour générer des tests automatiquement.

Elle reprend donc chaque brique du parcours et ajoute des tables qui représentent ses étapes de test avec des actions et résultats attendus. Marie a bien avancé sur la conception des tests : elle pourra présenter des extraits de ces tests demain à toute l'équipe et voir si cela est bien en phase avec la vision du reste de l'équipe.

Arrivée au bureau, Marie s'attaque à la conception des cas de tests de l'*US5* qui est prioritaire. Marie attrape donc son casque et se met une playlist qui la motive. Après le déjeuner, toute l'équipe se réunit pour la rétro : c'est le moment de faire le bilan de la semaine.

Sur le nouveau projet, l'équipe est hyper enthousiaste car cette première semaine s'est plutôt bien passée. D'un point de vue fonctionnel, les différents ateliers ont permis de bien affiner les besoins : des *US* sont conçues, elles sont détaillées par des exemples et des questions-réponses.

Marie a pu commencer la rédaction des tests et Lily et Pascal ont commencé ce matin les développements.

L'équipe se projette sur la semaine à venir et se donne pour objectif de faire une première démo au client vendredi prochain.

La rétro se termine par un tour de table pour recueillir les avis de chacun sur la semaine passée. Et tous plébiscitent le management visuel mis en place en remplacement de ce bon vieux tableur.
Les ateliers de réflexion sont aussi salués et font vraiment gagner du temps. “On fait beaucoup moins d'allers et retours et nos compréhensions sont plus rapides à aligner", précise Lily.

Marie et Paula quittent les locaux ensemble en discutant de ce qu’elles vont faire le weekend.

I.5- L'humain au cœur de l'outil, l'outil au cœur de l'humain

par Olivier Denoo

S'il est un sujet populaire au sein des technologies de l'information, c'est bien celui des outils. Il y en a pour tout et pour tout le monde : de la gestion des défauts, des versions et des configurations jusqu'à la gestion de la chaîne de valeurs (Value Stream Management) tout entière, de l'automatisation fonctionnelle à la performance en passant par la comparaison visuelle ; il n'y a qu'à choisir.

Statiques, dynamiques, analytiques, pilotés par des I.A. dernier cri ; sous-licence, Open Source, à la demande, au forfait…il y en a pour tous les goûts, tous les besoins et tous les budgets, au point d'y perdre ses repères.

Pourtant, lorsque d'aventure on aborde la question de l'humain, c'est bien souvent le silence ou l'incompréhension. Des outils, vraiment ?
Et pourtant, oui, ils existent bien ces **outils de l'humain**, même s'il faut bien concéder que leur approche est souvent différente, moins « hi-tech », moins dépendante des logiciels.
C'est sans doute une des raisons pour lesquelles ils sont plus méconnus dans le petit monde de l'IT, ou qu'ils sont à peine effleurés dans les cursus des écoles d'ingénieur trop exclusivement préoccupées par les matières technologiques ; un « oubli » d'autant plus regrettable que la plupart des approches modernes, notamment issues de la mouvance Agile, s'appuient très fortement sur le travail en équipe.

Il me paraissait donc crucial, à l'heure de publier cet ouvrage, d'éveiller l'attention des lecteurs sur quelques approches fondamentales (parmi tant d'autres), faciles à mettre en œuvre et qui apportent rapidement cette valeur si chère à l'Agilité.

1- Communication

S'il est un sujet récurrent dans la problématique des développements logiciels – et plus largement dans les relations humaines, en général, c'est bien la **communication**.
Nous savons depuis des années que plus de **40% des problèmes découverts au sein de nos logiciels proviennent d'exigences ambigües et de besoins mal formulés**[1] ; nous avons tous en tête cette balançoire dont le design évolue, avec plus ou moins de bonheur, au gré des étapes et des prétendus besoins des parties prenantes ; nous avons redistribué les rôles et les tâches, rebaptisé les exigences en User Stories pour mieux les afficher sur des murs de post-it ; nous avons reformalisé, virtualisé, modélisé, automatisé…mais avouons-le : sans compréhension mutuelle, sans validation par de vrais utilisateurs, point de salut !

[1] Tom Gilb ou encore B. Boehm

Déjà, dans les années 50, Schramm formalisait les règles de la communication dans son modèle inspiré de Shannon-Weaver :

– Toute communication implique un émetteur (qui envoie le message) et un récepteur (qui le reçoit, à qui le message est destiné).

– Le message est encodé par l'émetteur sur base de son champ d'expérience propre (son vécu, ses codes, son vocabulaire, ses valeurs…).

– Le message est décodé par le receveur sur base de son champ d'expérience, qui est, par nature, différent de celui de l'émetteur.

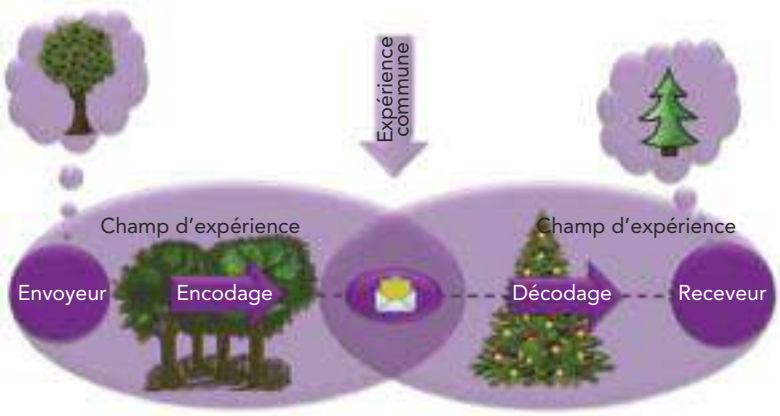
– Plus cette différence est grande et plus le message est altéré, incompris.

Ainsi, si je vous parle d'un arbre, par exemple, l'image qui s'impose à moi, celle de « mon » arbre est un modeste pommier au fond du verger de ma grand-mère…et voici que jaillit, sans crier gare, ma « madeleine de Proust » : ah ses tartes aux pommes au bon goût de beurre, parsemées de cannelle et de raisins de Corinthe !

Mais pour vous, peut-être, le mot « arbre » évoque les guirlandes de Noël, l'odeur de la résine, le bonheur de la surprise, du chocolat chaud, du papier que l'on froisse et des cadeaux sous le sapin…

Notre expérience commune se bornerait donc à un mot, à un concept - celui de l'arbre - qui se perd et se déforme dans ces champs d'expérience qui nous sont propres et qui sont radicalement différents.

Si nous devions donner corps à ce simple message, « arbre » ; si je prenais la posture du commanditaire et vous celle du maître d'œuvre ; si vous traduisiez ce message en Exigence ou User Story sans autre forme de validation… je doute fort que le résultat final passerait l'étape d'acceptation.



Modèle de communication de Schramm (1954)

Serions-nous donc condamnés à ne jamais nous comprendre, à errer sans cesse pour chercher la sortie de cette tour de Babel ? Non, bien heureusement !

Schramm lui-même apporta un élément de solution en faisant évoluer ce modèle linéaire vers un modèle circulaire, où les éléments envoyés et reçus sont (ré)interprétés et revalidés par les parties prenantes afin de s'assurer de la bonne compréhension du message.

Il suffit généralement de rephraser, de poser quelques questions judicieuses (Pourquoi ? Comment ?) et de procéder à quelques itérations pour lever la plupart des ambigüités. Dans la même optique, un formalisme strict (Gherkin, approche BDD…) et les techniques classiques d'élucidation aident à venir à bout des non-dits et des incompréhensions.

« **L'Agilité apporte la valeur plus rapidement** », lit-on souvent de nos jours.
C'est sans doute vrai, pour autant que cette notion fondamentale de « valeur » ait bien été validée par ceux à qui elle est destinée, à savoir les utilisateurs finaux.
Qu'elle soit perçue ou réelle, la valeur est difficile à cerner et reste largement dépendante des circonstances et du contexte.
Le message, par nature diffus et ambigu, émis par nos utilisateurs, toujours plus nombreux, toujours plus diversifiés, passe lui aussi par ces phases d'encodage, de décodage et d'interprétation qui font typiquement partie du champ d'expertise des analystes métier[3].
Si nous voulons offrir à nos utilisateurs la qualité qu'ils méritent, si nous avons l'intelligence de comprendre qu'elle est un différenciateur essentiel, il nous faut immanquablement quitter nos tours d'ivoire, descendre dans l'arène et confronter notre champ d'expérience au leur.

2- Organisation

L'humain a besoin de repères : Qui suis-je ? Où vais-je ?[4]
Il a besoin de guides, de garde-fous. Il doit donner du sens à sa vie, à son travail - Quel est mon rôle ? Quelle est ma valeur ajoutée ? - tout en restant, autant que faire se peut, en accord avec ses propres valeurs.

L'histoire suivante illustre bien cette problématique du sens, des valeurs et de la perspective :

« Un jour, je passai devant ce qui me semblait être une grande bâtisse en construction. Curieux, je m'arrêtai devant un maçon qui posait des briques et je lui demandai ce qu'il faisait.

– Je construis un mur, me répondit-il, non sans une certaine fierté.

Comme sa réponse me laissa sur ma faim, j'avançai sur le chantier et, remarquant un contremaître, je lui posai la même question :

– Je dirige la construction d'un cloître, me répondit-il, mais si vous voulez avoir une vue plus large, allez donc voir l'architecte, là-bas.

Quand, un peu plus loin, je posai la même question à l'architecte, affairé au beau milieu de ses plans, il me répondit, d'un air qui se voulait important :

– Je planifie et supervise la construction de la plus belle cathédrale que le pays aura jamais vue.

[2] Voir IREB et IQBBA
[3] Plutôt que de les reprendre dans ce chapitre, je renvoie le lecteur souhaitant approfondir le sujet aux techniques d'élucidation des exigences abondamment décrites dans les syllabi IREB CPRE et IQBBA.
[4] Dans quel état j'erre ? aurait ajouté le regretté Coluche

Alors que je m'apprêtais à quitter le chantier, maintenant que ma soif de savoir était étanchée, je croisai un prélat tout vêtu de rouge, qui allait grand train. Je ne pus m'empêcher de lui poser la même question et la réponse qu'il me fit donna encore un tout autre éclairage à la chose :

> *– J'entretiens la foi et perpétue l'œuvre de Dieu en bâtissant une maison pour accueillir ses fidèles »*

Si cette parabole moyenâgeuse illustre bien la hiérarchie des besoins et des exigences (ou des User Stories, Epics, Features ou Themes si vous préférez), elle nous éclaire aussi sur les notions de rôles ou de tâches propres à nos organisations.

L'Agilité et la transformation digitale de notre monde ont considérablement œuvré à modifier en profondeur les structures organisationnelles au sein des entreprises et des projets. En déplaçant le pouvoir vers des équipes pluridisciplinaires (ou cross-fonctionnelles) fonctionnant avec une large autonomie, en redéfinissant la répartition des rôles ou des tâches, en voulant réduire les inefficacités du passé, elles ont induit de nouveaux risques qui, s'ils ne sont pas correctement pris en compte, contribueront à créer de nouvelles problématiques.
Je ne puis qu'applaudir quand je lis, ou que j'entends : « **la Qualité est l'affaire de tous** ».

Pourtant, quelquefois, la beauté du slogan se heurte à la réalité du terrain.
Dans ces structures où, faute de formation et d'encadrement adéquats, plus personne ne se sent investi (individuellement), la qualité devient un murmure minoritaire vite étouffé… voire, en pratique, « l'affaire de personne ».

Car ne nous y trompons pas, l'humain, faut-il le répéter, a besoin de **sens**, il doit trouver de la **valeur** et, dans une certaine mesure, du **plaisir** dans ce qu'il fait. Lorsqu'une tâche nous rebute, lorsqu'elle est dévalorisée ou dévalorisante, lorsqu'elle ne contribue pas à notre épanouissement, nous avons une fâcheuse tendance à la négliger, à la laisser à d'autres, à procrastiner.

De la même manière, force est de constater que, bien souvent, les membres des équipes nouvellement constituées ont été choisis pour leurs **compétences fonctionnelles**, sans tenir aucun compte de leurs **compétences humaines** – j'entends par là les fameuses « **soft skills** » - alors que les équipes qui réussissent le mieux combinent harmonieusement **savoir-faire** et **savoir-être**.
Chacun d'entre nous a sa propre manière de se connecter au monde, de réagir face aux différentes situations auxquelles nous sommes exposés, avec nos forces et nos faiblesses.
Si nous sommes tous uniques à notre manière, nous pouvons toutefois nous retrouver peu ou prou dans des catégories plus larges, des typologies ou des profils, qui permettent de mieux appréhender notre valeur ajoutée, notre manière de réagir et de communiquer, pour le plus grand bénéfice de tous.

Diverses approches, plus ou moins sérieuses ou plus ou moins discutables, ont fleuri sur le sujet. De nombreux modèles en ont été tirés, comme **MBTI, Lohminger** ou encore l'**ennéagramme**, pour ne citer que ceux qui ont marqué ma propre expérience. Si l'on applique les nécessaires

garde-fous et la distanciation intellectuelle qui va de mise, chacun d'eux apporte son lot de connaissances de soi, de sa connexion au monde et aux autres ; chacun d'eux trace le chemin vers l'efficacité relationnelle et mène à une certaine sérénité.

Cette complémentarité nécessaire des compétences et des profils se retrouve également dans le modèle de **Belbin**, qui décrit ainsi neuf types principaux, avec leurs forces, leurs faiblesses, mais aussi dans une certaine mesure, le(s) moment(s) où ils apportent toute la plénitude de leur valeur.

Au travers de questionnaires d'évaluation assez simples, le modèle permet d'identifier les différents profils qui constituent l'équipe : ces visionnaires qui nous montrent le chemin avant de passer à autre chose, ces perfectionnistes qui veillent au souci du détail, ces coordinateurs qui s'assurent que le monde tourne rond quand d'autres prennent soin du bien-être de tous ou font la promotion du travail commun.

Bien que nous ne soyons pas cantonnés dans un seul rôle/type, et que ce dernier ne nous définisse pas entièrement[5] - heureusement d'ailleurs - nous avons néanmoins nos préférences, nos zones de confort et aussi nos aversions.

La valeur réside, selon l'approche, dans la diversité et dans la capacité de l'équipe à endosser les différents rôles au moment opportun. Plus l'équipe est complète et équilibrée et plus elle a de chances de mener son projet à bien[6].

– **Coordinateur** : diplomate, clarifie les objectifs et fait progresser les prises de décision
– **Soutien** : sociable, attentif aux autres et à leurs besoins
– **Promoteur** : communicatif, explore les opportunités, développe les contacts
– **Concepteur** : créatif, propose des idées et des solutions originales
– **Priseur** : stratégique, évalue et envisage toutes les options
– **Expert** : autonome, source de connaissance et de compétences
– **Propulseur** : dynamique, aime la pression pour surmonter les obstacles
– **Perfectionneur** : consciencieux, s'assure que le travail est fini, recherche les erreurs et omissions
– **Organisateur** : méthodique, transforme les idées en actions concrètes et pratiques

Bien que cette approche – commerciale, comme la plupart des autres, faut-il le souligner – n'est ni parfaite ni exempte de risques[7], elle a l'immense mérite de favoriser la connaissance de soi et des autres, de comprendre que nous ne sommes pas tous pareils, de nous aider à identifier et à embrasser nos différences pour mieux transcender le vivre-ensemble et les résultats.

3- Identification / compréhension

– Mais que veulent-ils au juste ?… Ils changent sans cesse d'avis.

– Mais, ce n'est pas du tout comme ça qu'il faut l'utiliser, ce n'est pas prévu ainsi…

– Ce truc ne marche pas…c'est infernal, j'ai beau appuyer, rien ne se passe… si je confirme ? Oui mais que se passera-t-il si je le fais ? Et si je ne le fais pas ?

[7] Notamment celui de la facilité que nous avons à « coller des étiquettes », forcément réductrices et qui nous collent à la peau avec les inévitables clichés qui les accompagnent

Combien de fois avons-nous lu ou entendu ces phrases qui résument si bien le fossé, le mur d'incompréhension qui sépare concepteurs et utilisateurs ?
Raccourcir les cycles de production, itérer à qui mieux mieux pour embrasser le changement, cela peut, certes, réduire la dissonance ; ces actions sont insuffisantes si elles ne s'accompagnent pas de mesures destinées à capter cette fameuse valeur dont tous se revendiquent, souvent à tort.

Connaissons-nous vraiment nos **utilisateurs** ? Et nos **produits/services** ?
A l'heure où le logiciel s'infiltre dans tous les aspects de nos vies, nos utilisateurs se diversifient, leurs profils deviennent plus flous. L'informatique ne peut plus se contenter d'être faite par les informaticiens, pour les informaticiens ; elle doit rencontrer le plus grand nombre, celui des quidams qui consomment les technologies sans en connaître les ressorts, les tenants et les aboutissants, celui des utilisateurs qui se moquent des langages, des frameworks et autres outils qui font notre quotidien.
« Il n'y a qu'à les éduquer, à leur apprendre ? »
Et pourquoi donc ? La plupart n'en ont cure. Devenez-vous un expert en médecine ou en dentisterie avant de prendre un rendez-vous médical ? Devez-vous être pilote pour prendre un avion ?

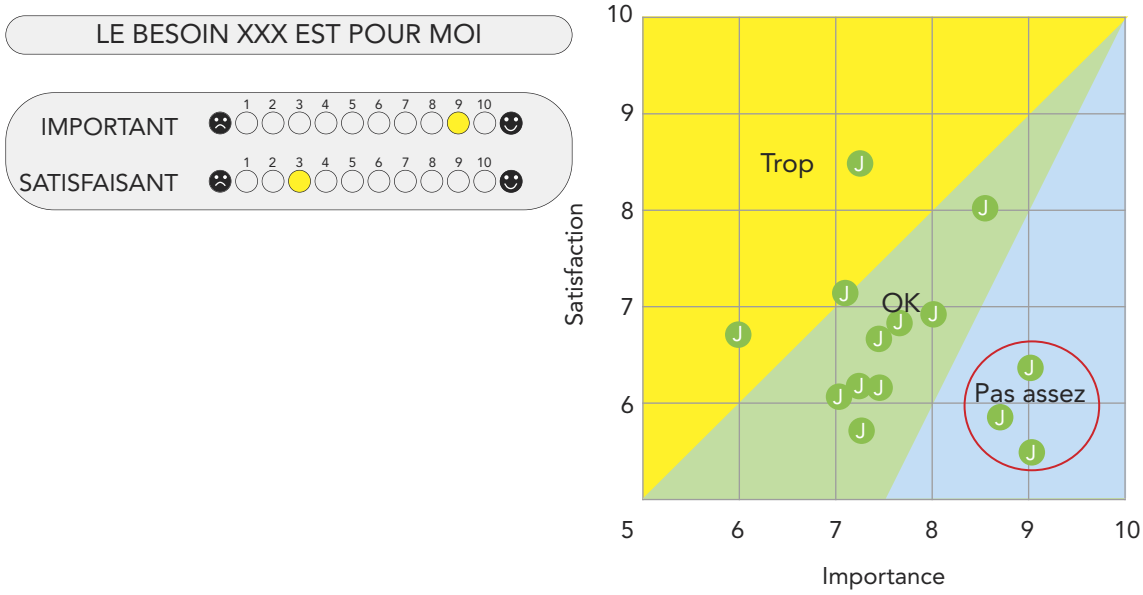
Une enquête de l'OCDE classait à peine 5% de la population testée dans la catégorie « *possède de très bonnes connaissances en IT* »[8] …ce qui signifiait en d'autres termes que ces personnes étaient capables d'accomplir des tâches basiques sur un traitement de texte ou de prendre un rendez-vous dans un application de messagerie ! Au temps pour les « *tout le monde sait cela* » et autres « *notre application est naturellement intuitive* » !

Il existe pourtant de nombreuses approches et outils, comme le **design participatif**, qui impliquent l'utilisateur dès les premières étapes du cycle de vie ; des questionnaires pour évaluer leur satisfaction, l'utilisabilité du produit ou du service, en cours de développement ou a posteriori ; la **carte d'empathie**[9] pour mieux les cerner… et si l'on ne peut vraiment pas impliquer des utilisateurs réels, au moins peut-on prendre le temps de les idéaliser, de les modéliser au préalable sous forme de **personae**[10].

L'identification et la priorisation des améliorations à apporter à un produit ou service peut souvent se résumer à un simple questionnaire de satisfaction mené avec soin. Pour autant, qu'un nombre restreint d'utilisateurs représentatifs - en proportion conforme à la réalité de l'usage (ou de la cible) - ait pu être identifié, et qu'on ait convaincu ces utilisateurs de donner leur avis en répondant au questionnaire[11], une simple matrice mettant en relation l'importance d'une fonctionnalité (exprimée sous forme de besoin et matérialisée par un point sur le graphe) et la satisfaction que l'on éprouve vis-à-vis de cette dernière en l'état actuel des choses, suffit à dégager les 3 grandes zones de « niveau de service »[12] :

[8] OCDE Skills Matter 2011-2015 – 5% population "highly skilled in IT" –215942 personnes testées dans 33 pays au total
[9] https://openclassrooms.com/fr/courses/5249081-gerez-un-projet-design-avec-plusieurs-methodes-de-ux-mapping/5718241-degrossissez-votre-sujet-avec-la-carte-d-empathie-empathy-map
[10] Il existe même de nombreux sites et outils permettant de modéliser des personae plus ou moins détaillés, et d'autres pour réaliser des questionnaires de satisfaction ou d'utilisabilité
[11] Cet aspect n'est pas trivial, dès lors que nous sommes constamment sollicités par toutes sortes de démarches. Le taux de réponse s'améliore toutefois avec la qualité de la relation-client, la qualité de l'écoute et de l'impact perçu de nos réponses sur la solution finale ou l'emploi d'éléments incitateurs (cadeau, rémunération, concours)
[12] Le lecteur remarquera que nous avons volontairement limité les dimensions de la grille aux valeurs supérieures à 5 pour des raisons de lisibilité.

1. Une zone où le service dépasse les attentes. Ici, le produit/service a tendance à « surjouer ». En fonction de la stratégie, cette zone permet soit de réaliser des économies en revenant à un niveau plus adapté, soit de constituer un facteur d'excitation (au sens de Kano[13]) qui agit comme un différenciateur positif.
2. Une zone centrale où les attentes sont correctement rencontrées par le produit/service. Le « curseur » y est bien positionné et doit être maintenu à ce niveau afin d'éviter toute insatisfaction des utilisateurs (en particulier pour les fortes valeurs de l'importance) liée à une éventuelle dégradation.
3. Une zone où le service est en-dessous des attentes. Cette zone correspond aux besoins non rencontrés, qui génèrent de fortes frustrations. C'est donc là qu'il faut agir en premier lieu en développant les améliorations nécessaires selon leur ordre d'importance.



4- Product Vision Board

Le **Product Vision Board**, dont le modèle a été développé par Roman Pichler[14]- illustré ici par un exemple de service de livraison de repas à domicile - est un outil permettant de visualiser rapidement un service ou un produit.

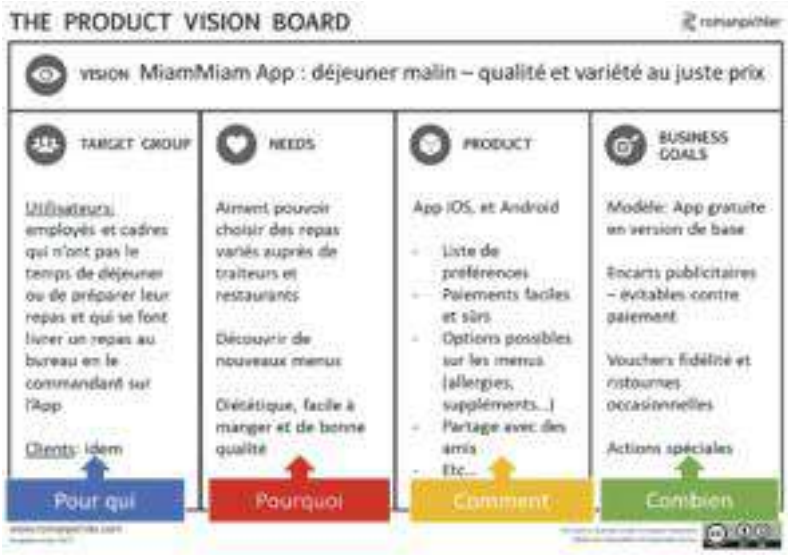
Son intérêt principal réside dans sa facilité de mise en œuvre. Quelques dizaines de minutes suffisent généralement à l'équipe pour obtenir un premier résultat satisfaisant.

Il favorise en outre grandement, pour tous les membres de l'équipe, la **compréhension du produit/service** ; ses caractéristiques générales, l'identification des tenants et aboutissants ainsi que son positionnement.

[13] Voir le syllabus IREB CPRE FL ou https://fr.wikipedia.org/wiki/Mod%C3%A8le_de_Kano#:~:text=Le%20mod%C3%A8le%20de%20Kano%20est,des%20concepts%20de%20valeur%20sym%C3%A9trique.[14] https://openclassrooms.com/fr/courses/5249081-gerez-un-projet-design-avec-plusieurs-methodes-de-ux-mapping/5718241-degrossissez-votre-sujet-avec-la-carte-d-empathie-empathy-map

[14] Je conseille au lecteur une visite sur le site dédié qui, outre des outils intéressants, présente de courtes vidéos explicatives pour mieux comprendre les limites et l'usage du Product Vision Board - https://www.romanpichler.com/tools/product-vision-board/

Il s'agit d'un outil **didactique et collaboratif** qui facilite les échanges et le questionnement et bénéficie à tous les membres de l'équipe, à des niveaux variés. Il permet notamment aux qualiticiens de mieux comprendre les enjeux, de constituer une première ossature pour leur stratégie de test et d'en dériver des approches pertinentes qui se focalisent sur « ce qui compte ».
Son positionnement, fortement orienté Métier, permet en effet de rapidement dégager la valeur, tout en la fixant au sein de l'outil.



5- La valeur du qualiticien

Il y a peu de temps, une discussion avec le DSI d'une structure de taille moyenne m'avait laissé pantois. Cette personne m'avait soutenu que ses services, fervents adeptes de l'Agilité, n'avaient pas besoin de testeurs car la qualité, chez eux, était intrinsèque au code. Il lui semblait tout aussi évident que la documentation était superflue, puisqu'un code bien écrit parle de lui-même.
La machine à remonter le temps existe bel et bien, je venais de remonter trente ans en arrière !
Il ne me restait qu'à prendre congé, non sans lui souhaiter bonne chance. Il en aura besoin.

Heureusement aujourd'hui, beaucoup comprennent mieux l'Agilité et savent aussi faire la nuance entre « *au lieu de* » et « *plutôt que* »[15]. Si les approches modernes du développement logiciel ont, en effet, considérablement bouleversé les organisations (et les organigrammes) et que le testeur « traditionnel » est forcé de quitter sa zone de confort pour se remettre en question, j'y vois tout d'abord une formidable opportunité, pour lui, d'élargir son périmètre, de se réinventer.

Le Testeur est mort, vive le Qualiticien !
De QC à QA ou de QC à QE il n'y a qu'un pas qu'il est (souvent) bon de franchir.
Mort au testeur, s'il le faut, et longue vie au Coach, au Quality Engineer ou à l'Assistant qualité… qu'importe le flacon, pourvu qu'il ait l'ivresse !

[15] En référence à la traduction du manifeste Agile et à son interprétation.

Il n'en reste pas moins que, très souvent, le « Qualiticien nouveau » doit prouver sa valeur, qu'il doit « se vendre » ; et comme cette vente est souvent synonyme de survie, il passe par une période de stress, surtout s'il peine à se faire une place dans les nouveaux organigrammes 2.0.

Une vente, dans son plus simple apparat, se compose de trois éléments essentiels :

1. **Un client** (qui va « acheter » ou « consommer » le produit ou service) – on parle ici des clients de la qualité, pas uniquement des clients finaux. La question à se poser est donc : « Pour qui est-ce que je teste ? », autrement dit : « Qui est intéressé par les résultats que je produis ? ».
2. **Un produit / service** – on parle ici des livrables du test et de la qualité, donc pas uniquement du produit fini. Les questions clés, à mettre en adéquation avec les réponses précédentes, sont : « Qu'est-ce que je produis ? », « Quelle est ma contribution à l'équipe ? ».
3. **Un contrat / un accord** (qui valide la transaction, lorsque les parties sont d'accord de conclure) – les éléments clés en la matière portent sur l'adéquation entre l'offre (telle qu'au point 2) et la demande (telle qu'au point 1) dans un contexte de marché (valorisation du produit / service). La question clé est : « Mon client est-il prêt à investir autant pour le service / produit que je lui offre ? ».

Les produits de la qualité ne diffèrent en rien des autres. Ils répondent à des règles connues depuis de nombreuses année. A une phase de maturité succède un lent déclin qui, si nous voulons éviter l'obsolescence ou la substitution[16], doit s'accompagner d'un renouveau. En d'autres termes, face au changement, il nous faut innover, comprendre et restaurer la valeur, écouter le marché, transformer les critiques en opportunités.

Une fois encore, l'usage des outils décrits précédemment permet d'y voir plus clair.
Ainsi, une approche dérivée de la carte d'empathie, une enquête menée auprès de nos clients - ceux qui consomment les produits du test - apportera, sans nul doute, les réponses aux questions que nous nous posons :

- Sommes-nous trop lents, trop chers, pas assez techniques, pas assez compétents, trop intransigeants, trop théoriques, trop peu flexibles ?
- Tout le monde peut-il faire ce que nous faisons ?
- Arrivons-nous trop tard ou voulons-nous arriver trop tôt dans le cycle de vie logiciel ?
- Sommes-nous des empêcheurs de tourner en rond ?
- Sommes-nous « les autres », « ceux d'en face qui ne sont pas comme nous » ?
- Délivrons-nous les bons produits / services aux bonnes personnes et avec le bon niveau de détail ?
- Nos clients en sont-ils satisfaits et dans quelle mesure ?
- Où sont les zones d'amélioration, les points forts, les points faibles, les facteurs d'excitation et les principaux blocages ?
- …/…

[16] Qui consisterait, dans le cas présent, à se passer purement et simplement de test, comme le DSI que j'évoquais plus haut, ou à les remplacer par autre chose.

Bien évidemment – c'est à espérer du moins – le Qualiticien reçoit aussi des critiques positives qui, tout comme les attentes non rencontrées, lui indiquent où est sa véritable valeur et ce que l'organisation attend de lui. S'agit-il :

- De l'information (sur l'état du projet, du produit, sur sa santé…).
- D'une aide à la décision (faut-il ou non aller en production, maintenant ou plus tard, faut-il tester plus ou mieux que tester d'abord ? …).
- Des garanties, une assurance (Vais-je bien dormir cette nuit ? Le site tiendra-t-il le choc ?, …).
- De fournir la qualité attendue plus vite et moins cher (mieux tester, mieux sélectionner, mieux automatiser…), autrement dit, de manière plus efficace.
- …/… ?

Un atelier cross-fonctionnel, centré sur un Product vision Board des produits et services de la qualité, permettra, sans aucun doute, de mieux cerner les livrables attendus, pour aboutir à **une proposition de valeur** formulée comme suit :

« POUR (cible, client)

QUI EST FRUSTRE(E) PAR (douleur identifiée sur la carte d'empathie)

NOTRE SOLUTION (produit / service)

OFFRE (capacité à résoudre le problème) »

Bien que les problématiques soient souvent les mêmes, chaque situation, chaque recette, chaque remède apporté est fondamentalement unique en soi en ce qu'il dépend pour une large part de l'organisation, de la culture et du contexte de l'entreprise.

Il appartient à chaque équipe de faire son examen de conscience afin de définir la solution la plus adaptée et la plus pertinente.

Il appartient à chacun d'entre nous de trouver et de prouver sa valeur pour apporter sa petite pierre au grand édifice.

PARTIE II
APPRÉHENDER UN MONDE DE PLUS EN PLUS COMPLEXE


TESTER UN LOGICIEL MULTICLOUD, MULTIUSAGE, MULTITERMINAL, MULTICANAL, MULTILANGUE ET MULTI-UTILISATEURS,
C'EST MULTICASSE-TÊTE !
Pchiii
SCHHHH
CRAC
PPPLLLLC
KRRR
F. COINTE

II.0- Un monde de plus en plus complexe

par Marc Hage Chahine

Nous sommes dans un monde de plus en plus complexe.

Ce nouveau monde est gouverné par un slogan qui pourrait être « toujours plus ! ». « Toujours plus » dans le sens où le nouveau monde nous offre :

- toujours plus de possibilités, avec un choix toujours plus grand, comme sur les plateformes de vidéos à la demande,
- toujours plus vite, avec des nouvelles ou des nouveautés qui en chassent d'autres, comme avec les versions de logiciels ou de téléphones ou encore les chaînes d'information en continu,
- toujours plus de puissance, avec une course effrénée à la technologie comme avec les différents écrans ou terminaux,
- toujours plus impressionnant, avec des exploits et des images sensationnelles nécessaires à la génération d'adrénaline.

C'est un fait, l'idée n'est pas de savoir si cette complexité est bonne ou mauvaise, notre monde est de plus en plus complexe… et il est évidemment nécessaire de s'y adapter, au risque de subir les conséquences de cette non-adaptation.

Dans ce monde toujours plus complexe, les éléments qui nous impactent sont également toujours plus nombreux. Cette évolution se reflète bien dans le monde du logiciel… D'ailleurs, on pourrait même aller jusqu'à dire que le logiciel est une des causes de cette accélération et complexification de notre monde.

1- La complexité des services numériques

Les logiciels, tout comme notre monde, sont de plus en plus complexes avec de nombreuses problématiques et variables qui sont intriquées.

Il n'y a encore que quelques décennies, les logiciels étaient réservés à quelques professionnels, qui travaillaient sur des ordinateurs qui tournaient sur des OS simples et à des « geek » qui jouaient sur des consoles de quelques bits (8 bits pour la NES).

Les logiciels étaient peu utilisés et dans des contextes assez peu variés.

Avec les années 80, les ordinateurs ont commencé à se démocratiser. Les OS, notamment windows, les ont rendus plus accessibles.

Internet a commencé à émerger à la fin des années 90. Cela a permis l'accès à de nombreuses informations à des millions de personnes.

Les premiers smartphones sont apparus dans les années 2000, rendant les environnements d'utilisation de logiciels et d'internet encore plus nombreux.

Les années 2000-2010 ont popularisé l'IoT, les applications, les plateformes de vidéos, les réseaux sociaux et la numérisation des services publics. Depuis, tout citoyen est devenu « dépendant » des services numériques.

Depuis le début des années 2020, on voit des utilisations concrètes de l'IA et une généralisation du Cloud.

Ces évolutions, nombreuses et rapides, ont sorti les services numériques de leur silo avec leur impact relatif, car avec peu de possibilités et peu de personnes potentiellement impactées. De même, les logiciels sont toujours plus interdépendants et je ne suis plus en capacité de donner un exemple de logiciel parfaitement autonome (même les jeux consoles sont mis à jour à distance !).

2- La nécessaire adaptation des testeurs à cette complexité croissante

Ces évolutions des services numériques ont entraîné une forte hausse des impacts et la nécessité de se concentrer sur des aspects non fonctionnels autrefois optionnels. En effet :

- Les logiciels impactent maintenant tout le monde.

Cela veut dire que ces services se doivent d'être utilisables par tous. Cela mène à des problématiques d'accessibilité pour les personnes en situation de handicap et d'utilisabilité afin d'être « instinctifs ». De même, les services numériques doivent fonctionner dans de nombreux environnements différents, que cela soit d'un point de vue type de connexion ou terminal d'utilisation.

- Les services numériques véhiculent maintenant des données confidentielles.

Cela entraîne la nécessité d'avoir des logiciels qui protègent suffisamment les données sensibles.

- Les services numériques sont disponibles sur de plus en plus de supports / terminaux.

Cela entraîne une contrainte forte sur l'adaptabilité, mais aussi la capacité à tester le logiciel sur son environnement de « production ». Ces environnements vont de l'enceinte connectée à un avion, en passant par un téléphone, un ordinateur ou même des machines de tri des déchets.

- L'infrastructure des services numériques est de plus en plus complexe.

Cela entraîne une nouvelle complexité ! Il y a une multiplication des APIs, des architectures en micro services, la mise en place de services sur le Cloud.

- Les services numériques doivent répondre toujours plus rapidement.

Cela entraîne la nécessité de savoir tester des temps de réponse dans de nombreuses conditions… Car, avouez-le, à part si vous y êtes contraints, vous ne fermez pas une page web si cette dernière ne s'est pas affichée correctement au bout de quelques secondes.

- Les services numériques utilisent de plus en plus de technologies.

Cela entraîne la nécessité de savoir s'adapter à ces contraintes… Je pense notamment à savoir tester l'IA qui n'a pas de réponses déterministes… Mais l'IA n'est que la partie émergée de

l'Iceberg. Comment tester du multimodal ? Comment tester des APIs ? Comment tester de la reconnaissance d'image pour des portiques de sécurité ? Les testeurs et plus généralement les équipes de développement se doivent d'apporter des réponses à ces questions.

3- Conclusion

Comme vous pouvez le constater, la complexité n'a jamais été aussi importante… et n'a jamais augmenté aussi vite. Les compétences nécessaires aux « QA » sont de plus en plus pointues et variées… Et il est nécessaire que ces derniers arrivent à s'adapter pour répondre aux nouveaux défis des services numériques.

II.1- Le RGPD et les tests de sécurité

par Yves Duchesne

Introduction

Le RGPD est une contrainte réglementaire à prendre en compte dans différents domaines. S'il concerne en premier lieu le domaine juridique, les obligations qu'il fait peser sur les organisations se concrétisent aussi par des mesures techniques. Cet article se propose de clarifier ce sujet, en rappelant ce qu'est le RGPD et les contraintes qu'il vous impose, mais également en questionnant la place des tests de sécurité dans le dispositif.

1- Le RGPD ?

Le RGPD, c'est quoi ? C'est un sujet qui a déjà été présenté de nombreuses fois dans de nombreux médias depuis l'irruption de cet acronyme dans nos vies numériques, le 25 mai 2018. Il signifie « Règlement Général pour la Protection des Données » -GDPR en anglais- et a été créé pour légiférer sur les pratiques relatives au traitement de données personnelles ; jusqu'ici, l'absence de règles claires avait permis la généralisation de pratiques délétères en la matière.

Ainsi, deux problèmes majeurs étaient constatés :

- De nombreux sites avaient pris l'habitude de collecter, de stocker et de revendre les données personnelles de ses utilisateurs, de façon abusive, en dehors de toute légitimité et de tout contrôle en la matière. Par exemple, il était anormal que des blogs culinaires collectent votre numéro de sécurité sociale.

- Par ailleurs, le niveau de sécurité de ces sites et les mesures techniques de protection des données personnelles qu'ils hébergeaient étaient largement insuffisants. Cela faisait peser un fort risque de compromission et de fuite de ces données, et cela a largement alimenté le marché noir en matière de courtage spécialisé sur les données personnelles.

Il y avait donc un manque global de maîtrise du traitement des données personnelles ; elles étaient collectées de façon sauvage, en dehors de tout contrôle, et elles ne faisaient pas l'objet d'une attention suffisante quant à leur sécurisation.

Par ailleurs, l'opacité autour de leur utilisation et de leur cession posait des problèmes de transparence de la vie publique, comme l'a mis en lumière le scandale autour de « Cambridge Analytica » : des données issues du réseau social Facebook avaient été transmises et utilisées lors de campagnes d'influence en ligne, dans le cadre des élections américaines.

Aussi, le Parlement Européen a voté le règlement du RGPD, qui s'applique en l'état aux différentes juridictions nationales des pays membres, afin de réguler les pratiques et d'offrir un meilleur contrôle sur les données personnelles, leur collecte, leur traitement, mais également d'apporter des garanties sur le niveau de sécurité de leur hébergement.

2- Les contraintes du RGPD

Depuis le 25 mai 2018, toutes les entreprises françaises et européennes sont donc soumises au RGPD et doivent respecter ce règlement.

Point important : le RGPD est un règlement européen ; cela signifie qu'il s'applique en l'état dans les différents états membres, à la différence d'une directive, qui doit être déclinée dans les différentes juridictions de ces états. La directive NIS, par exemple, fixe des objectifs qui doivent être implémentés et adaptés aux différents systèmes juridiques. Ainsi, le RGPD s'applique **exactement de la même manière** à toutes les entreprises qui en dépendent, **quelle que soit leur nationalité** et quel que soit le Droit national dont elles relèvent.

Le RGPD est un règlement complexe, qui demande à être pris en charge avec précision et peut nécessiter de se faire accompagner. Néanmoins, sa mise en place se fait selon plusieurs jalons clés :

Désignation d'un Délégué à la Protection des Données (DPD -DPO en anglais). Il s'agit du référent en interne des sujets relatifs à la protection des données. Il est en charge de la gouvernance des données personnelles au sein de votre organisation.

A ce titre, il doit informer, conseiller, alerter, etc., mais également répondre aux sollicitations externes. Il est à la fois le pilote interne et le correspondant externe.

Cartographie des données personnelles. Pour mener à bien les missions du DPD, le premier travail à déclencher est un inventaire des données personnelles qui sont traitées par votre organisation. Il débouche sur la création du registre des traitements de données, qui recense les données traitées et documente la légitimité du traitement.

Ce point essentiel permet de s'assurer de la nécessité pour votre organisation de collecter ces données personnelles et de justifier la présence de ce traitement. Il permet également de garantir la proportionnalité du traitement de données et de garantir qu'il n'est pas excessif, au regard de la finalité de cette collecte.

Evaluation des risques. Une fois les données identifiées et recensées au sein du registre de traitement, dans l'objectif de mettre en place la stratégie de sécurité de ces données, une analyse d'impact doit être réalisée. Il s'agit de l'AIPD (Analyse d'Impact relative à la Protection des Données), ou PIA (*Privacy Impact Analysis*) en anglais.

Cette réflexion initiale, essentielle dans la mise en œuvre de mesures de sécurité, est calquée sur l'analyse de risques, qui constitue la brique fondamentale pour toute démarche en cybersécurité. Elle consiste à mettre en perspective les biens sensibles de l'organisation et les menaces qui pèsent sur elle, afin d'identifier les scénarios de risque auxquels elle est confrontée et qu'elle devra prendre en compte pour définir sa stratégie de cybersécurité.

Dans le cas du RGPD, l'AIPD reprend cette méthodologie et se propose d'identifier les scénarios de risque pesant sur les données personnelles faisant l'objet d'un traitement, d'en évaluer les impacts et de mettre en place des mesures de sécurité permettant de les réduire.

Par exemple, il est possible de procéder à une anonymisation des données, afin de faire disparaître l'impact pour les personnes concernées, ou à une « pseudonymisation » des données, afin de réduire l'exposition des identités réelles.

Mise en place de procédures. La mise en œuvre du RGPD implique enfin de mettre en place des procédures relatives aux données personnelles, en particulier :

- Minimisation : il est nécessaire de mettre en place une politique visant à réduire au maximum les données collectées ; il s'agit d'une contrainte pesant sur votre organisation, mais c'est également l'opportunité de réduire sa surface d'exposition au RGPD et d'éviter au maximum les ennuis avec la CNIL ;

- Recueil du consentement : il est obligatoire d'informer les utilisateurs que leurs données personnelles vont faire l'objet d'un traitement et de recueillir leur accord pour qu'elles soient collectées ;

- Consultation et modification des données : les utilisateurs dont les données personnelles font l'objet d'un traitement bénéficient de droits d'accès à ces informations. Ils doivent pouvoir solliciter le DPD afin de prendre connaissance de ces données, mais également de les modifier, de les supprimer ou de retirer le consentement qu'ils ont précédemment donné ;

- Violation de données personnelles : une violation de données est l'accès illégitime avéré à des données personnelles. Il se produit lorsque les mesures de sécurité mises en place ont échoué à prémunir l'organisation d'une fuite de données. Les raisons peuvent être multiples : piratage, naturellement, mais aussi environnement mutualisé entre plusieurs clients ou maladresse lors d'une sauvegarde/migration de données, etc. Le cas échéant, il est nécessaire de définir des procédures de notification de la CNIL et des utilisateurs concernés par cette violation de données, afin que ces procédures puissent être déclenchées.

3- Les audits de sécurité

Dans le cadre du RGPD, l'AIPD a pour objectif de mettre en exergue les risques et les impacts relatifs aux données personnelles en prenant en compte les sources de menace et les différents scénarios qui peuvent se produire.

Ainsi, dans le cas d'une application exposée à un grand nombre d'utilisateurs, le risque de compromission (de piratage) doit être pris en compte et donnera lieu à un scénario de risque de fuite de données personnelles.

Pour réduire ces scénarios de risque, l'organisation doit pouvoir démontrer le bon niveau de sécurité de l'application, rendant un piratage improbable, et disposer d'éléments opposables permettant d'étayer ces allégations. Dans la plupart des cas, il n'est pas possible d'apporter d'éléments de cette nature tant que la sécurité de l'application n'a pas fait l'objet d'une expertise.

En l'espèce, l'intervention la plus indiquée est l'audit de sécurité, dont l'objectif est précisément d'évaluer le niveau de sécurité d'une application ou, plus généralement, d'une cible (SI, serveur, application, etc.).

C'est en cela que les tests sur la sécurité des applications occupent une place importante dans le dispositif RGPD mis en place par une organisation : ils permettent d'objectiver le bon niveau de sécurité d'une ou plusieurs applications, afin d'écarter les scénarios de risque basés sur un piratage. En l'absence d'éléments opposables et dans le cas d'un contrôle de la CNIL (dans le cadre d'une violation de données personnelles consécutive à une compromission, par exemple), elle pourra estimer que vous avez abusivement écarté ces scénarios et fait preuve d'une négligence coupable et de légèreté en la matière, ce qui peut mener à une sanction et une amende.

4- Quels audits ?

L'audit de sécurité est une intervention dont l'objectif général est d'évaluer le niveau de sécurité d'une cible (ici, une application). Il consiste à comparer le niveau de sécurité à un référentiel en réalisant des constats d'audit.

Mais derrière cette définition générale se cache une réalité multiple ; il existe en effet plusieurs méthodes pour réaliser cette évaluation du niveau de sécurité. En particulier, quatre types d'audit sont habituellement pratiqués et correspondent aux quatre portées techniques du référentiel PASSI (Prestataire d'Audit de la Sécurité des Systèmes d'Information, schéma de qualification des prestataires d'audit de cybersécurité proposé par l'ANSSI).

Le test d'intrusion est le principal type d'audit de sécurité et le plus connu de tous. Il s'agit du fameux *pentest* (mot valise, contraction de *penetration test*) auquel les développeurs et testeurs sont aujourd'hui habitués.

Pour évaluer le niveau de sécurité d'une application, ce test la confronte à des attaques informatiques ; il permet ainsi d'identifier les vulnérabilités présentes en déterminant quelles attaques ont fonctionné. Il se déroule habituellement en plusieurs phases séquentielles de :

- prise d'empreinte, afin de cartographier la surface d'exposition de la cible et d'identifier les technologies en présence (langages, bibliothèques logicielles, frame works, etc.) ;

- recherche de vulnérabilités, en tentant d'identifier des mauvaises pratiques (mauvaise implémentation, configuration dangereuse, mots de passe par défaut, etc.) ;
- exploitation de ces vulnérabilités, afin de démontrer la réalité effective de ces anomalies et d'évaluer leur impact sur la cible.

Les tests d'intrusion sont habituellement menés selon une méthodologie en boîte noire (lorsque l'auditeur ne dispose d'aucune information sur l'application), en boîte grise (lorsque l'auditeur dispose de comptes d'accès à l'application, afin d'adresser l'ensemble du périmètre fonctionnel) ou en boîte blanche (en fournissant à l'auditeur le code source de l'application, ainsi que toute la documentation associée).

Le test en boîte blanche est celui offrant le meilleur niveau d'analyse et la meilleure profondeur de test. Il permet d'accélérer la prise en main et la compréhension de l'application par l'auditeur, qui peut alors se concentrer sur la partie de l'audit ayant le plus de valeur ajoutée : la recherche et l'exploitation de vulnérabilités.

Les principales vertus du test d'intrusion sont son efficacité (donc son coût raisonnable) et sa capacité à mettre en exergue des impacts ; en exploitant les vulnérabilités, il permet de mesurer avec finesse leur impact et facilite le travail de priorisation et la construction du plan d'action correctif. En effet, ces plans d'action se basent habituellement sur des indicateurs comme l'exposition des vulnérabilités, la difficulté d'exploitation mais aussi (et surtout) l'impact sur le système.

En revanche, le test d'intrusion ne propose pas une grande complétude dans l'analyse ; il ne peut évaluer que les mécanismes externes de l'application, qui sont exposés aux utilisateurs. Il ne peut pas adresser les mécanismes internes de l'application. En particulier, il ne permet pas de juger de l'efficacité des mécanismes de protection des données (personnelles ou non), notamment des algorithmes de cryptographie qui ont été choisis et implémentés.

L'audit de code source est une alternative au test d'intrusion ; il sert les mêmes objectifs d'évaluation du niveau de sécurité d'une application mais utilise une méthode différente pour y parvenir. En effet, il va s'attacher à analyser les pratiques de développement et analyser le code source de l'application pour identifier des anomalies en matière de sécurité.

Ainsi, comparativement au test d'intrusion, l'audit de code source propose une intervention avec une meilleure profondeur d'analyse et une capacité à :

- identifier des mauvaises pratiques de développement constituant une dette technique de l'application, alors même qu'elles ne constituent pas de vulnérabilités exploitables, offrant aux développeurs l'opportunité de les corriger en avance de phase ;
- étudier la mécanique interne de l'application et évaluer leur conformité à l'état de l'art en la matière ; en particulier, les mécanismes de cryptographie et de protection des données peuvent être évalués efficacement.

En revanche, l'audit de code source n'a pas une bonne capacité à mesurer l'impact d'une vulnérabilité qu'il aurait permis d'identifier. En effet, l'exploitabilité d'une vulnérabilité est difficile à évaluer en étudiant uniquement le code source, car elle demande une compréhension très fine du fonctionnement des différentes briques technologiques (bibliothèques logicielles, frameworks, etc.), qui sont parfois complexes et évoluent dans le temps en fonction des versions successives.

Par ailleurs, la présence de composants tierces, non couverts par l'audit de code source, peuvent avoir une influence concrète sur l'exploitation des vulnérabilités et les rendre impossibles, malgré une présence théorique avérée dans le code (serveur web, reverse proxy, etc.).

L'audit d'architecture est une autre forme d'audit en boîte blanche, s'attachant à évaluer le niveau de sécurité d'une application en étudiant les choix techniques ayant présidé à sa conception. En effet, l'architecture d'une plateforme applicative, ou plus globalement d'un système d'information, a un impact fort sur son niveau de sécurité.

Plusieurs principes de conception confèreront une robustesse structurelle à l'application. A titre d'exemple, il est possible de citer les suivants :

- Le cloisonnement vise à séparer au maximum les différents composants entre eux et à les isoler les uns des autres, afin d'éviter qu'un attaquant puisse progresser et/ou rebondir au sein du système d'information.
- Le zoning a pour objectif de définir des niveaux de sécurité associés aux différentes portions du système d'information, afin de définir une politique de filtrage efficace, permettant d'empêcher la progression d'un attaquant en son sein.
- Le filtrage des flux permet de réduire au maximum les surfaces d'attaque des différentes zones d'un système d'information, afin de ne laisser passer que les flux essentiels, en privilégiant les flux d'une zone sécurisée vers une zone de moindre sécurité.
- Les protocoles utilisés, qu'ils soient réseaux ou applicatifs, doivent également être questionnés, afin de s'assurer qu'ils intègrent bien les fonctions élémentaires de sécurité (notamment chiffrement et authentification des flux).

Ainsi, cet audit offre une analyse en profondeur du niveau de sécurité structurelle d'une plateforme applicative et de son système d'information.

L'audit de configuration constitue un excellent complément à l'audit d'architecture ; il prolonge l'analyse du niveau de sécurité d'une plateforme en étudiant le paramétrage de ses composants essentiels, afin de s'assurer de l'absence de configurations dangereuses (désactivation de mesures de sécurité, présence de mots de passe par défaut, utilisation de versions de protocoles non sécurisés, etc.).

Comparativement à l'audit d'architecture, qui va évaluer le niveau de sécurité structurel d'une plateforme applicative, l'audit de configuration va permettre de prolonger la vérification et de s'assurer que les différentes briques qui le composent sont bien paramétrées.

Pour cela, les configurations sont d'abord extraites et exportées, puis analysées et comparées à des référentiels faisant autorité en la matière (notamment les guides de l'ANSSI, du NIST et du CIS).

Cet audit de configuration est traditionnellement mené sur la base d'un échantillonnage réalisé sur la plateforme analysée, en isolant les composants critiques qui peuvent être identifiés (pare-feux, serveurs, proxys, bases de données, etc.).

Conclusion

Pour auditer une application de façon pertinente dans le cadre d'une démarche RGPD, il est nécessaire d'opter pour une stratégie d'audit permettant de servir ses objectifs et de répondre aux contraintes qu'il impose. Ainsi, il faudra opter pour une méthodologie offrant la possibilité de juger des mesures de sécurité mises en œuvre pour protéger les données personnelles.

Or, la protection des données personnelles au sein d'une application relève principalement de deux choses :

- L'absence de vulnérabilités techniques permettant une fuite de données (injection SQL, notamment) ;
- La présence de mécanismes de protection robustes (en particulier, utilisation de cryptographie adaptée).

Si les tests d'intrusion offrent une bonne capacité à évaluer le premier aspect, ils se montreront en revanche relativement peu efficaces pour adresser le second, qui procède de la mécanique interne à la plateforme applicative.

Par conséquent, il paraît intéressant de privilégier un audit en boîte blanche, proposant une profondeur d'analyse de nature à vous offrir une bonne visibilité, à la fois sur les bonnes pratiques de développement permettant de vous prémunir des vulnérabilités techniques, mais également sur les mécanismes internes de protection des données, en particulier sur les mécanismes cryptographiques que votre application met en œuvre.

Ainsi, l'audit de code source peut apparaître comme un candidat intéressant, qu'il faut idéalement privilégier au test d'intrusion, car il offrira une meilleure capacité à adresser les contraintes du RGPD.

II.2- L'importance du test manuel dans l'accessibilité numérique

par Antonio Ferreira

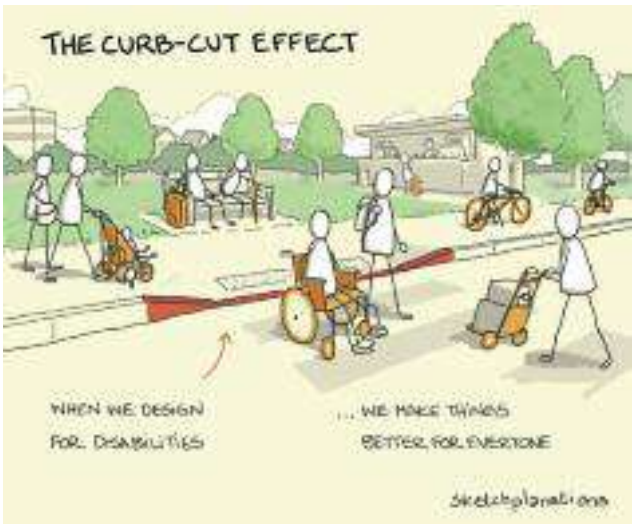
1- Introduction à l'accessibilité numérique.

L'accessibilité numérique est un pilier fondamental dans la construction d'un monde inclusif auquel chaque individu, quelles que soient ses capacités, peut accéder et utiliser les ressources disponibles sur Internet et les technologies associées. Elle vise à éliminer les barrières existantes qui limitent l'accès aux informations et services en ligne, dans une ère ou être connecté est presque obligatoire.

Imaginons un instant la situation d'une personne aveugle au plus fort de la pandémie de COVID-19. Alors que le monde se tournait massivement vers le digital pour travailler, apprendre, faire ses courses, se divertir, etc., cette personne se retrouvait isolée dans un moment où la connexion numérique était un lien vital avec le monde extérieur.

On peut réfléchir de façon similaire à d'autres types de handicaps, comme la mobilité réduite, des troubles cognitifs, des troubles auditifs, etc. L'Organisation Mondiale de la Santé estime à 1.3 milliards le nombre de personnes en situation de handicap dans le monde ; cette statistique souligne l'importance sociale de l'accessibilité numérique. De nombreuses juridictions à travers le monde ont reconnu l'urgence d'un Web accessible, en mettant en place des cadres légaux pour en assurer la mise en œuvre. Ces lois et régulations, inspirées par les principes des Web Content Accessibility Guidelines (WCAG), ou le Référentiel Général d'Amélioration de l'Accessibilité (RGAA) en France, visent à créer un monde numérique universellement accessible.

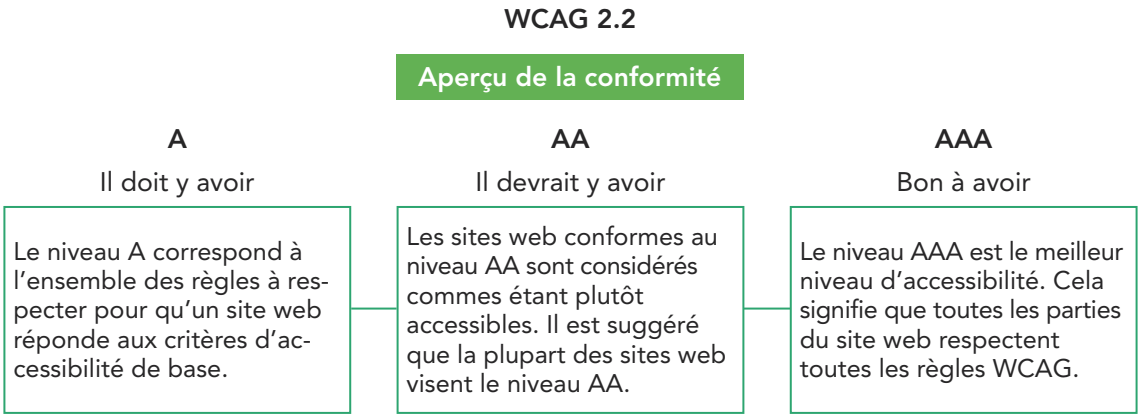
En œuvrant à rendre le web et les technologies numériques plus accessibles, nous ouvrons également la porte à un marché potentiel immense. Toutefois, les avantages de l'accessibilité sont illustrés par le concept de l'effet "bateaux de trottoir".



Initialement conçues pour faciliter le déplacement des personnes en fauteuil roulant, les bordures abaissées ont prouvé leur utilité pour une vaste gamme d'usagers, y compris les parents avec poussettes, les voyageurs avec valises et les enfants à vélo. Ce principe se vérifie également dans le numérique : en rendant les sites web et applications accessibles, on améliore leur usabilité pour tous, créant une expérience utilisateur plus riche et plus inclusive.

2- Comment débuter dans le monde du test d'accessibilité ?

Lorsqu'on débute dans le monde de l'accessibilité numérique, il peut sembler difficile de savoir par où commencer. Heureusement, des ressources sur YouTube comme la série "A11ycasts with Rob Dodson" pour les anglophones ou la série "Accessibilité numérique par DesignGouv" fournissent un point d'entrée offrant un équilibre entre les principes fondamentaux et leur application pratique en leçons courtes.
Au-delà de YouTube, il est essentiel d'explorer les ressources introductives fournies par le WCAG et le RGAA mentionnés dans l'introduction. Ces directives constituent le fondement de l'accessibilité web partout dans le monde et sont considérées, comme disent les Anglais, comme "the gold standard". Le "WCAG Quick Reference" est un outil précieux pour naviguer à travers les différents critères, offrant une vue d'ensemble claire et pratique. WCAG est organisé en 3 niveaux de conformité :



La législation américaine (American with Disabilities Act) et européenne (European Accessibility Directive et European Accessibility Act) se basent sur le niveau AA. Il faut aussi noter que selon WCAG « il n'est pas recommandé d'exiger la conformité au niveau AAA en tant que politique générale pour des sites entiers, car il n'est pas possible de satisfaire à tous les critères de réussite du niveau AAA pour certains contenus. »

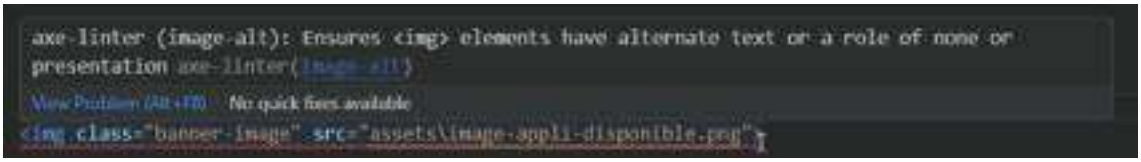
En France, on a un référentiel d'application pour répertorier toutes les bonnes pratiques correspondant aux niveaux d'accessibilité WCAG A et WCAG AA : le Référentiel Général d'Amélioration de l'Accessibilité (RGAA) avec 106 critères. Il faut savoir que chaque critère de la RGAA fait référence à un ou plusieurs critères RGAA. Toujours en France, en plus de la RGAA, nous sommes obligés d'avoir une déclaration d'accessibilité, de tester au moins certaines pages importantes de notre site/app et d'avoir un schéma pluriannuel d'accessibilité.

3- Le test automatique

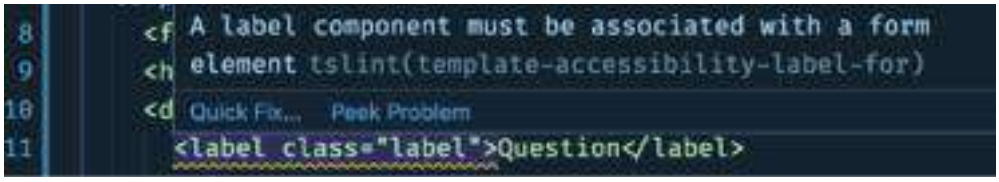
Lorsque l'on aborde le test de l'accessibilité numérique, les outils automatiques jouent un rôle crucial, bien qu'ils ne soient qu'une partie de la solution globale. Ces outils sont conçus pour scanner le code des pages web et les applications afin d'identifier les problèmes d'accessibilité qui peuvent être détectés de manière algorithmique. Cependant, il est important de noter que malgré leur efficacité, les tests automatiques couvrent généralement entre 10% et 50% des problèmes d'accessibilité dans le meilleur des cas. Souvent, plus le site est simple, plus la couverture des tests automatiques est importante. Ces limites soulignent la nécessité de compléter les tests automatiques avec des évaluations manuelles pour une couverture complète. On a deux types de logiciels automatiques :

3.1 Linters de code : un premier niveau d'assurance qualité. Ils permettent de scanner le HTML directement dans l'éditeur de code pour anticiper les problèmes directement dans le développement.

- Exemple de Axe-linter extension sur VS Code, quand il détecte qu'une image de texte alternatif manque, il signale cela avec une barre rouge impossible à ignorer :



- Exemple du package npm typescript-eslint qui détecte automatiquement un label qui n'est pas associé à un champ de formulaire avec un warning.



3.2 Audits automatiques d'un site web ou URL : des logiciels qui vont scanner une URL complète dans un browser. Souvent, elles se basent sur l'HTML et le CSS pour vérifier des points tels que :

- le contraste de couleur entre éléments,
- la structure du HTML pour vérifier qu'on n'a pas des éléments manquants,
- l'utilisation des attributs HTML WAI-ARIA valides,
- le texte alternatif des images,
- le titre et la langue des pages,
- le zoom sur des pages,
- l'utilisation des landmarks pour le contenu principal de la page,
- la taille des éléments interactifs.

Quelques exemples d'outils populaires :

- **Axe-Core :** proposé par Deque, Axe-Core est probablement la solution opensource la plus populaire intégrable dans CI/CD et aussi dans les navigateurs via des extensions. Elle est la base de Google Lighthouse et Microsoft Accessibility Insights.

- **Google Lighthouse** : intégré dans les outils de développement de Chrome, Lighthouse fournit un rapport sur l'accessibilité (utilisant Axe-Core), mais aussi sur la performance, les bonnes pratiques et plus encore. Un de ses principaux avantages est sa facilité d'utilisation et son intégration directe dans le navigateur, ainsi qu'une intégration facile en CI/CD avec le package npm lighthouse-ci.

- **Microsoft A11y Insights** : mon préféré pour les débutants, il intègre les tests automatiques de Axe-Core, mais aussi des instructions pour faire du test manuel avec des outils qui aideront un testeur non formé à débuter en accessibilité.

- **Wave** : similaire à l'extension Axe-Core, il offre une visualisation intuitive des problèmes d'accessibilité directement sur la page testée, facilitant ainsi l'identification et la compréhension des erreurs à corriger. A mon avis, son seul point négatif est qu'en général, il a plus de faux positifs que les trois autres .

- **Google A11y Scanner (Android)** : un scanner d'accessibilité pour des applications mobiles beaucoup moins complet que Axe-Core mais souvent un des seuls outils pour analyser les apps natives.

L'avantage principal des tests automatiques est leur rapidité, ils permettent de scanner de grandes quantités de contenu pour nous donner rapidement un avis global de notre app. De plus, certains peuvent être intégrés dans les environnements de développement CI/CD, permettant ainsi de détecter les problèmes d'accessibilité de façon continue.

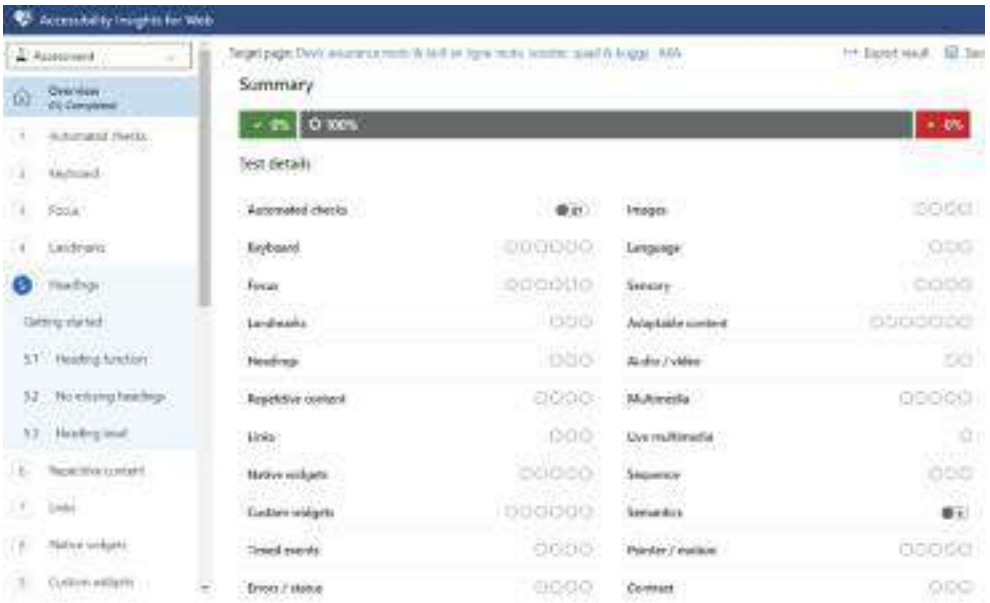
Malgré ces points forts, il est crucial de comprendre que les tests automatiques n'analysent pas le vrai comportement d'un lecteur d'écran sur un browser, ils se basent quasi exclusivement sur l'analyse du HTML et le CSS. De plus, on a souvent des pages dynamiques qui rajoutent du HTML dans la page selon notre interaction, par exemple avec une modale qui n'apparaît que quand on appuie sur un bouton et souvent ces éléments sont oubliés par le test automatique. Finalement, en attendant des avances dans l'IA, les tests automatiques ne peuvent pas encore identifier les problèmes qui requièrent une évaluation subjective, comme la pertinence des textes alternatifs pour les images ou la logique de navigation.

4- Faire un test manuel

Le test manuel joue un rôle indispensable dans l'évaluation de l'accessibilité numérique, complétant les lacunes laissées par les outils automatiques. Pour faire du test manuel, il nous faudra un lecteur d'écran (par exemple NVDA) et un outil pour mesurer le contraste entre deux couleurs (par exemple « Color Contrast Analyzer » de The Paciello Group).

Pour avoir une grille de test complète, il suffit d'aller sur le « WCAG Quick Reference » (disponible en anglais), la grille de test RGAA (disponible en français) ou bien l'outil « va11ydette » de Orange (disponible en français et anglais pour WCAG et RGAA).

Pour avoir une grille de test guidé pour les débutants, avec des explications et des outils complémentaires pour aider dans le test manuel, je recommande fortement l'application Microsoft Accessibility Insights. Voici la façon dont y est structuré le test par sujet, avec les tests automatiques de Axe-Core intégrés dans le logiciel dans la section « automated-checks » :



Les bases de NVDA : NVDA (Non Visual Desktop Access) est un lecteur d'écran gratuit et open source qui permet aux utilisateurs aveugles ou malvoyants d'utiliser un logiciel informatique. Pendant le test manuel, on utilise aussi la fonctionnalité « speech viewer », ça nous permet de voir en output textuel ce que la voix du lecteur est en train de lire. Pour commencer à utiliser NVDA pour tester l'accessibilité, il est essentiel de se familiariser avec quelques commandes de base :

- Touche Tab : permet de naviguer à travers les éléments interactifs d'une page, tels que les liens, les boutons, les checkbox et les champs de formulaire.
- Flèches Haut et Bas : utilisées pour lire le contenu textuel, permettant une navigation ligne par ligne à travers la page.

5- Test d'un site automatique + manuel

Bien que nous n'ayons pas le temps de couvrir toute la grille de test dans cet article, nous pouvons nous concentrer sur les outils de test automatiques et manuels pour identifier les problèmes d'accessibilité les plus fréquents. Faisons un test sur un site dédié à l'assurance des deux-roues pour analyser son accessibilité. On va se focaliser sur le top 10 des critères WCAG, qui représentent 80% des problèmes détectés sur un site quand on fait un audit d'accessibilité ;

on va aussi lancer un test automatique et avec un peu de chance, il détectera des bugs dans notre top 10 et quelques bugs supplémentaires. On illustrera 1 exemple par critère.

En commençant par notre test automatique, Axe-Core a trouvé 27 problèmes, néanmoins, on pourrait les regrouper en 7 problèmes uniques : 3 problèmes de couleur, 1 langue de la page non définie, 1 alt text manquant, 1 usage de couleur incorrect, 1 problème de zoom :

Pour le test automatique, on peut voir la façon dont le logiciel détecte correctement plusieurs des problèmes de contraste de couleurs (WCAG 1.4.3). Plus concrètement, le texte gris #C5C5C5 « Moto », « Scooter » et « Quad » avec un fond blanc a un contraste de 1.72 : 1. On voit bien que ces 3 occurrences sont comptées comme 3 bugs différents mais en réalité, c'en est un seul.

De même, pour les textes grisés sur la page, on compte 14 issues du texte gris #A0A0A0 sur fond blanc, en réalité on pourrait les regrouper dans 1 seul bug. Même chose pour les 3 occurrences du texte blanc sur le fond bleu clair qu'on pourrait regrouper en 1 seul bug :

⚠ C'est pour cette raison qu'il faut faire attention au marketing de ces logiciels, car ce genre de comportement permet de gonfler les chiffres et les pourcentages de bugs trouvés. Ce n'est pas techniquement incorrect de dire qu'on a trouvé 20 occurrences du même problème mais en réalité, si on ouvre 20 tickets JIRA pour régler le même type de bug, il se pourrait que l'équipe

de développement, par désespoir, envisage d'engager la mafia pour nous éliminer rapidement. Maintenant qu'on connaît les faiblesses de l'automatisme, on peut aborder la partie manuelle avec notre top 10 :

- WCAG 1.1.1 (Contenu non textuel) : assurez-vous que tout contenu non textuel (images, graphiques, vidéos) ait un texte alternatif pertinent pour que les lecteurs d'écran puissent le lire. Les éléments décoratifs devraient être cachés au lecteur. Le test automatique a bien détecté l'erreur sur les 3 images des motos (un linter l'aurait détecté aussi, comme on l'a vu précédemment).

- WCAG 1.3.1 (Information et relations) : l'information sémantique présentée visuellement doit être la même pour un lecteur d'écran. Souvent, cela échoue parce que des éléments présentés visuellement sous forme de liste ne sont pas annoncés par le lecteur comme une liste. Un autre exemple classique qu'on trouve sur le site: un des titres de la page n'est pas annoncé comme titre par le lecteur parce qu'il est codé comme un paragraphe <p> dans le code :

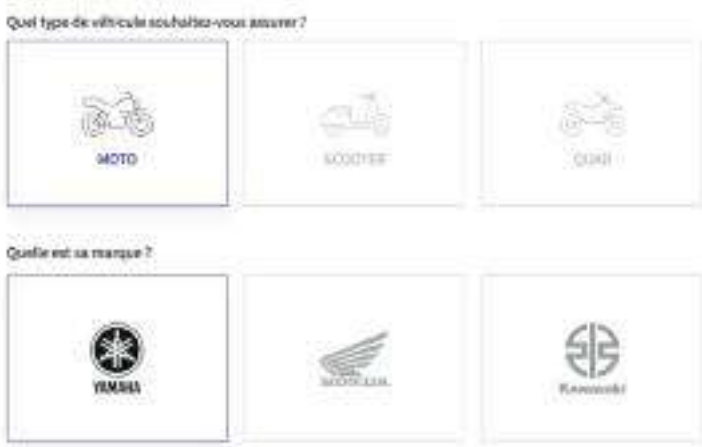
- WCAG 1.4.3 (Contraste minimum) : le texte et les éléments de l'interface utilisateur doivent présenter un contraste de couleur de 4.5 : 1 pour le texte normal et 3 : 1 pour le texte large. Le test automatique a bien détecté l'erreur.

- WCAG 1.4.11 (Contraste non textuel) : les informations visuelles requises pour identifier les composants et les états de l'interface utilisateur doivent avoir un contraste de 3 : 1. Par exemple, un icon button d'une maison pour aller à la page d'accueil doit avoir un contraste 3 : 1 avec le fond d'écran. Ici, cela échoue car l'indicateur qui indique le focus du clavier sur l'élément « moto » est trop léger :

• WCAG 2.1.1 (Clavier) : tout contenu doit être accessible via le clavier. On utilise la touche TAB pour bouger entre les éléments interactifs du site (boutons, liens, checkbox, etc.), les FLECHES pour naviguer dans certains composants plus complexes comme des tab groups et ENTER et SPACE pour activer des éléments. Courage pour arriver à sélectionner Kawasaki dans la liste avec les flèches ou tab :

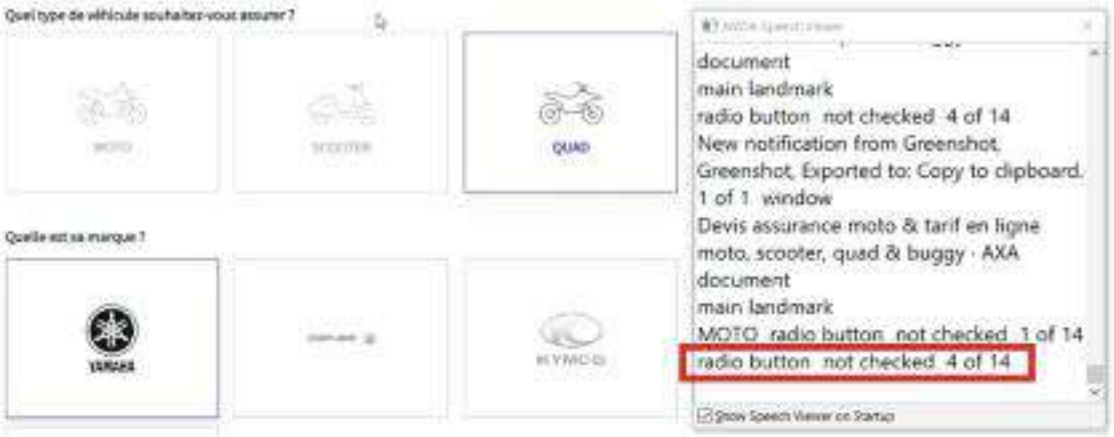
• WCAG 2.4.3 (Ordre de focalisation) : l'ordre dans lequel les éléments reçoivent le focus clavier doit être logique et intuitif. Par exemple, dans le site, on ne peut pas naviguer en arrière :

• WCAG 2.4.7 (Visibilité du focus) : l'utilisateur doit savoir à tout moment quel élément a le focus clavier. Ici, une fois qu'on a sélectionné une moto et une marque, si on navigue en arrière, le focus du clavier n'est pas distingué. Ici normalement, je suis sur « MOTO » mais je pourrais penser que je suis sur « YAMAHA ».



• WCAG 3.3.1 (Identification des erreurs) : les erreurs de saisie générées par l'utilisateur doivent être identifiées et expliquées à l'utilisateur de manière textuelle. Si on oublie de sélectionner notre véhicule, l'erreur nous informe que la réponse à la question est obligatoire. En revanche, si on oublie de sélectionner le modèle, on n'a aucun message d'erreur et cela compte comme failure du 3.3.1.

• WCAG 4.1.2 (Nom, rôle, valeur) : chaque élément interactif doit avoir un nom (un label), un rôle (button, link, checkbox, etc.), et une valeur (checked/not checked, disabled, selected, etc.) assignés pour que les lecteurs d'écran puissent transmettre cette information aux utilisateurs. Dans cet exemple, il manque le label d'un radio button.



• WCAG 4.1.3 (Messages d'état) : les messages d'état (comme les succès ou les erreurs de formulaire) doivent être présentés de manière à être perçus par les lecteurs d'écran, sans nécessiter le focus de l'utilisateur. Dans notre cas, l'erreur identifiée « vous n'avez pas répondu a cette question » n'est pas annoncé automatiquement au lecteur d'écran quand on appuie/active le bouton « Continuer ».

6- Conclusion

En conclusion, bien que l'automatisation des tests d'accessibilité soit cruciale pour identifier rapidement les problèmes détectables algorithmiquement, l'importance de l'évaluation manuelle par des experts en accessibilité reste primordiale. Je suis quelqu'un d'optimiste et ma vision du test d'accessibilité dans le futur est celle d'une automatisation à 100%.
Pour cela, il faudrait tester avec un vrai lecteur d'écran et interpréter la sémantique et les interactions du clavier avec une IA.

L'IA, avec ses capacités d'apprentissage et d'adaptation, pourrait aussi offrir de nouvelles méthodes pour comprendre et répondre aux besoins d'accessibilité.

II.3- Quality Engineering : le développement d'un système antifragile

par Antoine CRASKE

Dans un monde de plus en plus complexe, la pérennité des entreprises dépend de la résilience et de la capacité d'innovation de leur système de production logicielle.

Une offre supérieure à la demande, une innovation accélérée, une compétition accrue : autant de facteurs poussant les entreprises à se réinventer en continu. Sans exception, l'ensemble des acteurs sont soumis à une pression de transformation digitale dont le logiciel est la pièce centrale, représentant de nombreux défis pour apporter une valeur pérenne.

La construction logicielle consiste à planifier, spécifier, construire, délivrer et opérer des changements logiciels supportant un nombre grandissant de fonctions. Pour rester compétitives, les organisations nécessitent toujours plus de rapidité dans l'adaptation de leur proposition de valeur et poussent ces cycles d'itérations dans leurs retranchements.

1- Les points d'inflexion du Quality Engineering

Notre écosystème de plus en plus complexe augmente le degré d'incertitude et de volatilité auxquelles les organisations doivent faire face. Des chocs de plus en plus fréquents et importants émanant de l'environnement, la compétition ou la société impactent négativement la performance d'une entreprise qui ne peut pas résister.

Les ressources viennent également à manquer, la fragilité des acteurs augmente. Les organisations font face à une pénurie de talents avec 30% de recrutements restant vacants et 43% des collaborateurs ne se projetant pas à plus de 6-8 mois dans leur entreprise[1]. Les financements et pouvoir d'achat sont en baisse, augmentant la compétition sur les ressources.

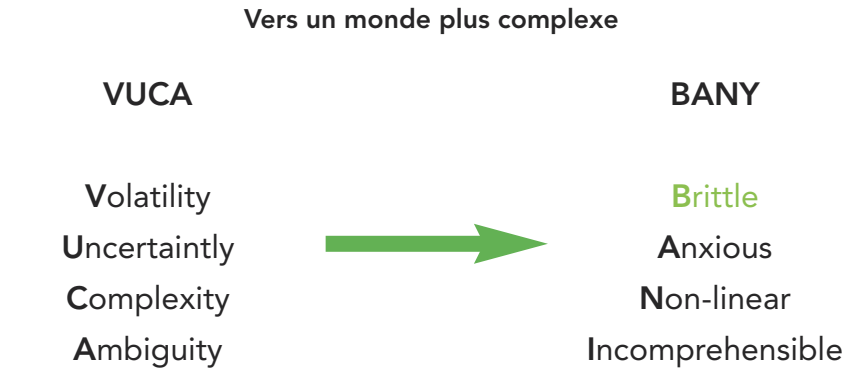


Figure 1 : Notre écosystème subit des changements profonds de complexité et de fragilité.[2]

[1] 2023. *The State of Organizations.* McKinsey & Company.
[2] Jeroen Kraaijenbrink. 2022. *What BANI Really Means.* Forbes.

En recherche de vitesse, nombre d'entreprises investissent dans des démarches de fluidification de leurs processus (e.g. Agilité, DevOps). Néanmoins, une accélération "à tout prix" sans des fondations robustes augmente le risque de défauts et la dette technique, deux facteurs qui finissent par faire perdre des clients et ralentir l'ensemble de l'organisation.

Les organisations doivent donc pouvoir développer des capacités internes pour réadapter rapidement leurs processus, résister aux chocs et les transformer en opportunités. Ces fondations permettent ensuite d'adapter leur proposition de valeur dans les segments les plus porteurs, en fonction des évolutions de l'écosystème.

2- Qu'est-ce que le Quality Engineering

Dans un contexte ultra-compétitif où l'adaptation rapide est nécessaire, les entreprises peuvent difficilement ralentir pour prendre le temps de la qualité. Elles ont donc besoin d'accélérer tout en maîtrisant la qualité, une équation qui semble difficile à résoudre dans un paradigme historique opposant vitesse et qualité.

Le Quality Engineering, quant à lui, repose sur le paradigme de la qualité totale où la vitesse est une conséquence d'un système optimisé. Le mantra "Construire mieux pour construire plus vite" résume l'essence de l'approche systémique développant le système de production logicielle pour maximiser la valeur, la vitesse et l'efficience des entreprises.

95% des organisations sont trop fragiles

*"95% des organisations annoncent des transformations **sans avoir les capacités digitales ou technologiques**"*
McKinsey & Compagny, Statte of Organizations 2023

Figure 2 : Le Quality Engineering développe le système de production logicielle dans sa globalité.

L'image souvent partagée du Quality Engineering est celle de l'intégration de technologies dans la chaîne de développement logiciel. Cette perspective doit être précisée par le besoin (i) d'intégration de la priorisation à la mesure de valeur en production, (ii) d'aligner les processus avec l'architecture technologique et organisationnelle et (iii) d'animer les acteurs au centre de la production logicielle pour développer une culture qualité intégrée.

Le Quality Engineering s'appuie sur trois domaines particuliers pour structurer et améliorer le système de production logicielle : la pensée systémique *(i.e. system-thinking*), l'architecture (à tous niveaux, de l'entreprise à la technologie) et les pratiques du Lean (des fondamentaux, jusqu'au DevOps et à l'Agilité) – L'objectif : développer un système anti-fragile.

3- Le développement d'un système antifragile

L'antifragilité est une propriété de systèmes qui arrivent à résister et à se développer sous des facteurs de stress[3]. C'est le niveau de maturité le plus élevé de résistance aux chocs, qui démarre à un niveau fragile (le système casse), pour évoluer vers de la robustesse (le système récupère) jusqu'à la résilience (le système résiste).

Un système antifragile se distingue par sa capacité d'adaptation continue en étant soumis à des stress continus. Une organisation bénéficiant d'un système de production logicielle antifragile peut réadapter ses offres plus rapidement, résorber les chocs d'incidents et capter le maximum de valeur temporaire face à ses concurrents.

Modèle de maturité du Quality Engineering

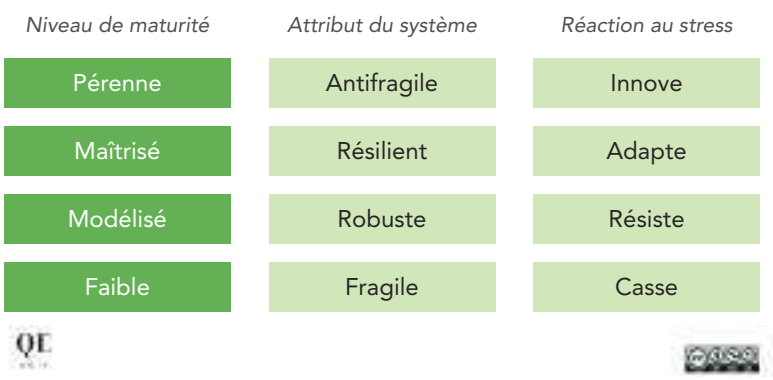


Figure 3 : Le modèle de maturité du Quality Engineering, de fragile à anti-fragile.

La définition du niveau de fragilité d'un système de production logicielle passe par l'évaluation de domaines clés de performance : entreprise, organisation, flux, fiabilité, efficience. Chacun de ces domaines définit des intervalles de performance observés dans les entreprises s'appuyant sur le digital pour rester compétitifs.

Le Quality Engineering définit la structure du système de production logicielle dans les 5 zones fonctionnelles de MAMOS (*Methods, Architecture, Management, Organisation, Skills*) qui comportent 10 capacités de production chacune. Les processus et technologies à intégrer sont modélisés ainsi que les capacités de support à développer et aligner dans le système.

[3] Taleb, Nassim Nicholas. 2013. *Antifragile*. Harlow, England: Penguin Books.
Antoine CRASKE, Rémi Dewitte. 2022. On Defining Quality Engineering. QE Unit.
Antoine CRASKE, MAMOS & QE Unit Blog. QE Unit.

4- Quelles implications pour les ingénieurs logiciels ?

Le Quality Engineering apporte un modèle de transformation holistique. Pour des entreprises déployant des démarches d'Agilité ou de DevOps, il permet de mieux prioriser les domaines à développer en fonction des enjeux de l'organisation et d'aligner processus, organisation et technologique pour optimiser la performance de bout en bout.

Pour les ingénieurs logiciels, cela signifie plus de transversalité et de renouveau pour rester à jour dans un écosystème en rapide adaptation. La capacité à collaborer avec des profils différents est clé pour fluidifier les chaînes de valeur et mettre en place des solutions globalement optimales. Les *soft skills* de collaboration, d'analyse critique et de résolutions de problèmes sont donc de plus en plus importants.

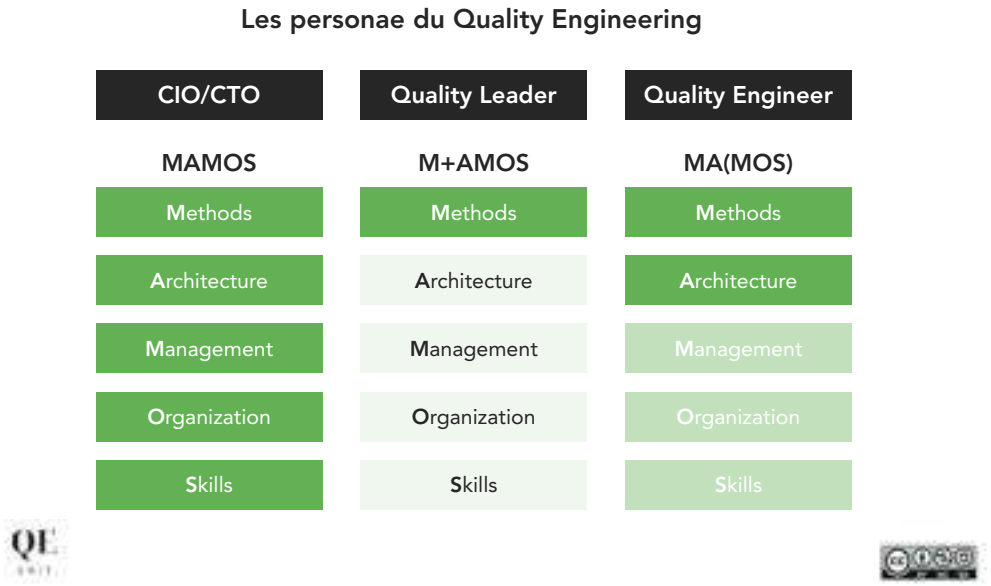


Figure 4 : Les personae et périmètres d'intervention du Quality Engineering.

Concrètement, les CTOs doivent développer leur compréhension du système dans sa globalité et orchestrer des changements structurels. Les leaders qualité doivent agir plus en transverse, du *Quality Advisor au QE Manager* pour développer des pratiques de *Shift-Left, Shift-Right* ou de *Platform Engineering*. Les profils d'automaticiens doivent aussi renforcer leurs *soft skills* pour mieux collaborer en transverse et apporter plus de valeur.

Ce besoin d'agir hors de la zone de confort dans un écosystème dynamique nécessite des individus à la capacité de renouveau continu. Les pratiques et technologies continuant à évoluer rapidement vers plus d'automatisation incitent chacun à investir dans son développement de compétences pour devenir un profil T-shape jusqu'à Comb-shape. Les leaders organisationnels ont également un rôle à jouer pour favoriser ce contexte d'apprentissage.

5- Comment aller vers le Quality Engineering

Un prérequis fondamental au Quality Engineering est un alignement des parties prenantes sur le besoin d'une approche systémique. Ce sujet doit être porté au plus haut niveau de sponsoring de l'organisation jusqu'au CIO/CTO pour assurer une démarche à fort impact. Des gains locaux peuvent être réalisés par des équipes intéressées, mais auront du mal à perdurer et résoudre des problèmes en dehors de leur périmètre.

Les signaux avant-coureurs poussant à une prise de conscience sont liés à la fragilité du système. Ils peuvent être le constat d'une faible valeur livrée malgré des compétences, en raison des retards récurrents malgré la planification et la disponibilité de l'outillage, ou une complexité organisationnelle impactant la productivité des équipes. Certaines organisations doivent subir un choc majeur et, une fois en phase de récupération, décident d'adresser la fragilité de leur système de production logicielle.

Une démarche de Quality Engineering démarre par l'identification et la priorisation des enjeux clés à adresser pour les utilisateurs du système de production logicielle. Ces enjeux se retrouvent en lien avec le modèle de maturité, corrélés à des indicateurs de performance d'entreprise comme le NPS, des indicateurs de livraison d'*Accelerate* ou d'efficience des ressources. Cette première étape réalisée permet de consolider les problématiques systémiques du système de production, pour lequel il faut définir les améliorations attendues en termes de métriques (*i.e. outputs*) et d'indicateurs de performance (*i.e. outcomes*).

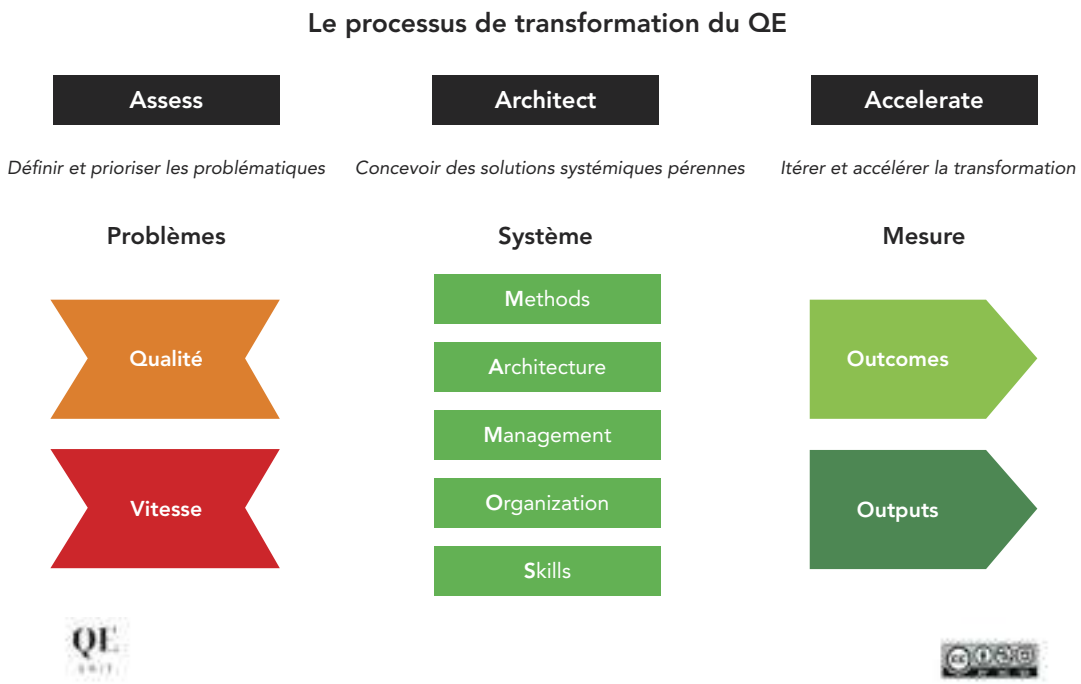


Figure 5 : L'approche itérative de transformation AAA.

Les échanges avec les différents interlocuteurs permettent de capturer le contexte organisationnel et la perspective des acteurs par leur connaissance du système. Leur expérience dans l'organisation permet de remonter des points structurants quant à la dynamique des problèmes rencontrés. Il est donc utile à cette étape de consolider le ou les flux de construction logicielle avec leurs indicateurs clés de performance, l'architecture technologique et la structure organisationnelle existante et ciblée.

L'étape suivante consiste à prendre le recul nécessaire pour diagnostiquer correctement les causes et renforcements sous-jacents à l'origine des problèmes. D'origine systémique, ils nécessitent des solutions systémiques pour en venir à bout, évitant les effets d'annonces et symptômes qui apparaissent pendant quelques mois. À cet effet, le Quality Engineering repose sur l'architecture de processus systématiques alignés avec les capacités de production logicielle, garantissant les prérequis et la pérennité des changements.

Ces évolutions doivent être réalisées avec un impératif de transformation pour mettre en place les changements sous quelques semaines. Un périmètre focus doit être défini pour tester les hypothèses d'améliorations de manière incrémentale et rapidement adapter les solutions par l'expérimentation. Les métriques et indicateurs de performance sont ici utilisés pour valider l'effective amélioration de la situation par l'adaptation du système. Une fois validées, les solutions peuvent être plus largement déployées par des dispositifs d'accélération.

6- L'antifragilité pour faire face á la complexité

Notre monde de plus en plus complexe et incertain renforce la pertinence pour les organisations d'investir dans leur système de production logicielle. L'investissement dans leur résilience et capacité d'adaptation par le logiciel leur permet de développer un avantage compétitif durable d'antifragilité face à leurs concurrents.

Le Quality Engineering rassemble les acteurs de la production logicielle pour focaliser leurs efforts dans les changements à plus fort impact. Ces transformations vont au-delà des processus ou de la seule technologie. C'est bien l'architecture de l'ensemble du système de production qui permet de fluidifier les chaînes de valeur et d'y aligner ses capacités.

Comme toute transformation, les premières itérations nécessitent des efforts pour initier la dynamique. Mais une fois lancé, c'est une dynamique organisationnelle d'amélioration continue qui s'enclenche pour développer un système de production antifragile. Et avec le temps, la meilleure connaissance du système accélère les cycles d'amélioration.

Nous pouvons difficilement influer sur la complexité croissante de notre monde, mais nous pouvons directement agir pour développer l'antifragilité de nos systèmes de production. Cela ne nécessite pas de position ou d'outillage particulier - cela passe par la capacité à changer notre perspective de la qualité.

II.4- Validation et cloud

par Frédéric Assante Di Capillo

Introduction

L'apparition du cloud dans le monde de l'informatique a bouleversé les habitudes des développeurs, notamment par l'apparition de nouveaux concepts :

- SaaS → Software as a Service.
- PaaS → Platform as a Service.
- IaaS → Infrastructure as a Service.
- XaaS → Toutes sortes de déclinaison de services.

Mais aussi par l'usage de nouveaux outils :

- Docker.
- Ansible (JSON & SSH).
- Puppet.
- Jenkins.
- Etc.

Les usages, pour le commun des mortels, peuvent se voir dans les applications distribuées (ex Suite Office), dans les objets connectés ou encore dans la vidéo à la demande. Pour les entreprises, beaucoup d'entre elles utilisent le cloud via les suites informatiques et de plus en plus de données et de logiciels sont proposés sous ce format. Nous pouvons prendre l'exemple d'Octane qui est proposé en solution Saas pour les entreprises ayant besoin d'un outil ALM.

Le cloud (là où tout a commencé !!). Hameau de la Creuse.

1- Histoire et acteurs

Contrairement à ce que nous pouvons croire, la technologie cloud n'est pas récente. Les mainframes, proposés dès les années 1950, peuvent être considérés comme les ancêtres du cloud. C'est la popularisation d'Internet et l'apparition des grands hébergeurs du WEB comme Google ou Amazon qui ont popularisé le terme de « Cloud Computing ».
Il est à noter que les premières applications vraiment déployées dans le « Cloud Computing » (souvent associées au WEB 2.0) sont issues des gestionnaires de mails, des outils collaboratifs, des CRM (Customer Relationship Management), des outils de développement et enfin des outils de gestion des tests (ce qui nous intéresse tout particulièrement !).

De nombreux acteurs se partagent le « gâteau » du « Cloud Computing ». La taille du marché est estimée, en cours d'année 2022, à environ 200 Milliard de $. Nous pouvons les classer par leur part de marchés détenues en 2022[1].

Fournisseur de services d'infrastructures cloud	Parts de marché 2022
Amazon Web Services	34%
Azure	21%
Google Cloud	10%
Alibaba Cloud	5%
IBM Cloud	4%
Salesforce	3%
Tencent Cloud	3%
Oracle Cloud	2%
Autres fournisseurs (OVH, Orange Business Services, Scaleway, T-systems, 3D Outscale, etc.)	18%

2- Avantages et Inconvénients

2.1- Avantages

Les principaux avantages du « Cloud Computing » sont les suivants :

- Une économie importante, notamment par l'utilisation à la demande des différentes applications proposées (vue utilisateur).
- Une mise à jour « transparente » des applications car elles ne sont pas sur les sites des clients mais sur des serveurs fournisseurs, plus de longues installations nécessaires pour les nouvelles versions (vue utilisateur).

[3] (en) « Infographic: Amazon Leads $200-Billion Cloud Market [archive] », sur *Statista Infographics*

- Une gestion des pics de demande facilitée par la possibilité intégrée d'activer des serveurs supplémentaires sur une période donnée (vue fournisseur).
- Une séparation des métiers entre les fournisseurs d'applications et les hébergeurs dans l'esprit de « chacun son métier » (vue fournisseur).

Bien d'autres avantages peuvent être cités, mais ce n'est pas le but de cet article.

2.2- Inconvénients

Les principaux inconvénients sont les suivants :

- Un coup écologique très (trop ?) élevé. Les fermes de serveurs consomment énormément, surtout en termes de refroidissement des serveurs avec un usage démesuré de la climatisation.
- Une perte de maîtrise des environnements d'hébergement des entreprises, donc une dépendance aux hébergeurs de plus en plus accrue.
- Une dépendance à la qualité des réseaux Internet /Ethernet. Avec un risque de perte financière élevée en cas de coupure prolongée.
- Une sécurisation des données plus risquées, car elles sont hébergées chez un tiers.
- Un risque accru de piratage et de perte des données sensibles.
- Une réversibilité des processus non maîtrisée.

Les quatre derniers points seront abordés dans le chapitre concernant la validation du cloud.

3- Validation dans et du cloud

Lorsque l'on parle de validation dans et du cloud, deux concepts sont à prendre en compte :

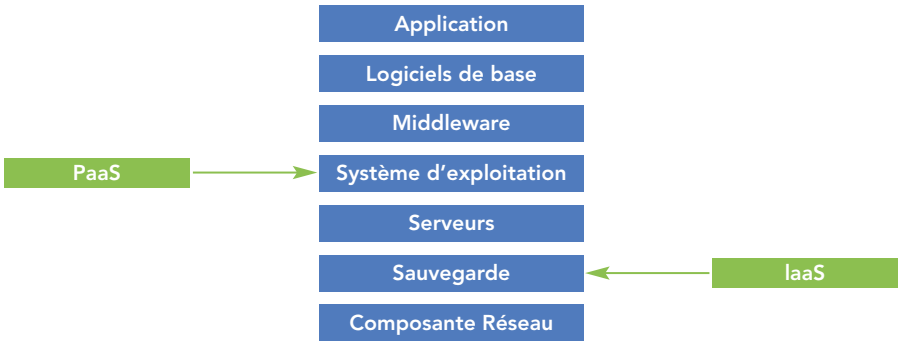
- Valider **dans** le cloud
- Valider **le** cloud (ou plutôt valider les applications dans le cloud).

3.1 Validation dans le cloud

Les applications gérées à travers le cloud ont un gros avantage, c'est qu'elles offrent un panel d'outils permettant de faciliter la validation. Nous pouvons mettre en avant deux modèles :

IaaS (Infrastructure as a Service) : permettant de déporter la gestion des Infrastructure vers le fournisseur. Par là même, nous pouvons imaginer déporter la gestion des Infrastructures de tests vers ce même fournisseur.

Paas (Platform as a Service) : permettant de disposer d'une plateforme à volonté, donc de créeer des plateformes de tests dès que l'on en a besoin et uniquement pendant les phases de validation.



Cependant, il ne faut pas oublier que ces services imposent certaines contraintes. L'aspect initial est, évidemment, la limitation des coûts pour les infrastructures de tests. En revanche, de nouvelles contraintes apparaissent :

- **Notion de gestion contractuelle.** Au même titre que pour une gestion « classique », il faut anticiper les besoins en termes de structure et de connectivité. En revanche, la différence fondamentale vient de la contractualisation de ces environnements avec un tiers acteur. Lors d'une gestion interne, il est toujours plus facile de « rattraper » les manques au fil de l'eau et de « forcer » pour disposer de notre environnement de tests en temps et en heure. Dans le cas d'un contrat externe, si les éléments de la plateforme de validation ne sont pas, ou peu, définis en amont, nous risquons fortement de nous retrouver avec un Add-On à ce même contrat, avec des coups non planifiés à prendre en compte. Cela peut même mettre en péril le démarrage de la phase de tests, en fonction de la disponibilité du provider pour délivrer cette nouvelle plateforme de tests.
- **Notion de data.** Les données permettant la validation sont toujours un élément clé de cette phase. Autant nous pouvons avoir une maîtrise de nos données en interne et procéder à une anonymisation pour respecter les normes telles que RGPD, autant cela doit être prévu en amont pour disposer d'un jeu de données en mode « cloud ». Des séances de travail avec le provider sont nécessaires pour s'assurer de la disponibilité de ces données. Cela peut être une copie de la production avec anonymisation, création de données de tests de toutes pièces ou génération aléatoire de données. En tous les cas, une anticipation de cette demande doit être prise en compte pour les mêmes raisons que précédemment.
- **Notion de connectivité**. La connexion vers d'autres systèmes peut être gérée de plusieurs manières. Nous pouvons décider de nous connecter à des environnements de tests des tierces parties ; simuler les connectivités externes ou « ghoster » ces liens. À l'instar de ce qui a été explicité précédemment, cela devra être discuté en amont du projet avec le fournisseur de solution cloud pour établir ces connexions vers des tierces parties.

Les trois points précédents sont les plus courants à aborder dans un concept de validation dans le cloud. Il est évident que cette liste n'est pas exhaustive mais que, selon les particularités de chacun, il faudra prévoir tous les points nécessitant des attentions particulières à la rédaction du contrat entre le fournisseur de solution cloud et vous.

3.2- Validation du cloud

La validation des applications hébergées dans le cloud ne diffère pas drastiquement de ce que nous pouvons faire pour les applications dites « classiques ». Les mêmes types de tests sont à faire :

- Tests fonctionnels
- Tests non fonctionnels
- Tests d'utilisation
- Tests de performance
- Etc.

En revanche, il faut amplifier (ou créer) certaines catégories de tests qui deviennent essentiels pour une application hébergée dans le cloud. Nous devons donc avoir :

- Des tests de sécurité :

Ce point est capital car la sécurité est un des problèmes majeurs des applications hébergées dans le cloud. Il faut donc vérifier que l'intégrité des données et leur sécurisation soient bien assurées. Sans rentrer dans le détail, il existe plusieurs types de sécurisation des données en fonction des normes que vous devez suivre (RGPG, PCI DSS, etc.). Nous pouvons citer :

- **Evaluation des vulnérabilités et des risques sécuritaires**, classés généralement en 3 catégories (low, medium, critique). Cela consiste à inspecter le code et à vérifier la fiabilité par rapport aux vulnérabilités connues et répertoriées (de nombreux outils existant peuvent vous aider pour la détection de ces éléments).

- **Audit de sécurité**, qui permet de vérifier, à l'aide d'un auditeur externe, que le code est à un haut niveau de robustesse face aux menaces.

- **Tests de pénétration**, connu aussi sous le nom de « piratage éthique », permettant de vérifier qu'il est très difficile de pénétrer votre application (il est bon d'avoir conscience qu'une protection totale est illusoire).

- **Tests de réactivité**, permettant de tester le comportement de l'application en cas de piratage, afin de limiter au maximum les impacts (fermeture des accès, isolation des parties infectées, déconnexion, etc.).

- Des tests de performance :

Même si ce type de tests n'est pas implicitement lié aux applications hébergées dans le cloud, il est à valider avec une plus grande attention que pour une application « classique ». En effet, les performances sont directement liées au type d'hébergement et au contrat passé avec le fournisseur. Votre application peut avoir de très bonnes performances mais être grandement pénalisée dans son environnement cloud. Ces tests sont très difficilement réalisables sur des plateformes de tests, il faut donc souvent les réaliser en production sur une plateforme dédiée (voir chapitre précédent sur la gestion des plateformes pour la validation). Il est important de vérifier que la plateforme résiste aux pics de charge prévus (ou non !) et que l'augmentation de capacité automatique (souvent prévue au contrat avec le fournisseur) anticipe bien les demandes de charges.

• Des tests de disponibilité :

Ces tests permettent de vérifier que l'accès à votre application correspond au cahier des charges signé avec le fournisseur de la solution d'hébergement dans le cloud. Ces tests ne peuvent pas être ponctuels, mais ils consistent à mettre en place un monitoring de votre application pour détecter toute perte de connection.

• Des tests de récupération :

Qui dit application hébergée dans le cloud dit passage par un ensemble de « canaux » ayant, par définition, des chances de s'arrêter de façon impromptue (les tests précédents ne couvrent que la disponibilité, pas la partie « recovery » !). Il faut donc s'assurer qu'en cas de coupure dans la chaîne d'accès, notre application réagit correctement, peut récupérer ces données lors de la reconnexion et ne perd pas des informations en route. Un point sensible étant le fait de « polluer » des informations, qui deviendraient inutilisables par la suite. L'exemple type est d'empêcher un client d'avoir une coupure réseau quelconque, et de ne plus pouvoir récupérer ce client car il est dans un statut intermédiaire non géré par notre application.

Nous pouvons aussi citer les tests suivants :

- Tests de compatibilité, si vous désirez être potentiellement hébergé sur différents clouds).
- Tests de portabilité, si vous décidez de quitter le monde du cloud et de retourner à un mode dit « classique ».
- Tests écologiques, permettant de connaître le comportement de votre application en termes de consommation de ressources (ici essentiellement de l'énergie).
- Etc.

Conclusion

Le cloud a révolutionné notre usage de l'informatique mais aussi notre façon de développer nos produits/applications. Les phases de validation doivent s'adapter à ces changements. Même si nous n'avons pas de grandes révolutions de nos processus de qualification, il se dégage quelques grands axes pour être le plus efficient possible :

- Anticiper les phases de validation pour prévoir nos besoins lors des contrats avec les fournisseurs de solution cloud.
- Bénéficier des avantages du cloud par l'utilisation des Paas pour gérer nos campagnes.
- Augmenter nos couvertures qualitatives, avec l'introduction (ou l'accroissement) de nouveaux types de tests.

La gestion dans le cloud, si elle est maîtrisée, peut être un plus pour les phases de qualification.

II.5- En route vers l'ingénierie de la performance

par Philippe Bridel

La performance applicative est un ensemble de propriétés de l'application indiquant son bon comportement, tant en qualité de service qu'en consommation de ressources.
Dès lors, les tests de performance applicatifs regroupent un ensemble de pratiques permettant de vérifier ce bon comportement lorsque l'application est sollicitée. On pense naturellement aux tests des temps de réponse. Mais le terme est plus générique et on peut inclure les tests de charge, de capacité, d'endurance, de robustesse, de résilience…

La performance est un élément essentiel pour un opérateur comme Orange, notamment à travers la qualité de service perçue par le client. La satisfaction de celui-ci est un facteur de différenciation par rapport à nos concurrents. La performance, élément différentiant, est considérée par Google comme étant la fonctionnalité la plus importante du produit ou du service et doit être prise en compte dès sa phase de conception.

Depuis quelques années, la manière dont on appréhende la performance applicative est en constante évolution. L'arrivée de l'Agilité, du DevOps, du cloud et des applications cloud natives a engendré une transformation du métier et de l'approche de la performance. Nous sommes passés du test de la performance à l'ingénierie de la performance.

1- De quoi s'agit-il ? Quels sont les impacts sur les équipes et les projets ?

Dans un projet en cycle en V, les tests de performance arrivent en général en fin de cycle, avant la mise en production, ce qui peut s'avérer tardif si des problèmes d'architecture ou de conception sont constatés. On voit désormais de plus en plus de projets basés sur une méthode agile. Cependant, si la partie performance n'est pas intégrée dans les différents sprints, on va retomber sur le problème précédent. Pour obtenir un feedback plus rapide, on va opérer un mouvement vers l'amont et prendre en considération les questions de performance dès le début du projet : c'est le mouvement dit « shift-left ».

Le mouvement « shift-right » consiste à accompagner de manière rapprochée l'application côté opérations. La combinaison des deux mouvements permet d'effectuer du test de performance en continu, dans la chaîne CI/CD. L'ingénierie de la performance est composée de ces deux mouvements, qui bouclent pour s'alimenter mutuellement, dans un mouvement DevOps. Tout ceci a bien sûr un impact sur l'organisation. L'ingénieur Performance n'attend plus la fin du projet pour intervenir. Il a toute sa place dès les premiers sprints.

Selon les projets, il pourra être plus ou moins présent dans les différents sprints. Lorsqu'il est absent, un relai devra être trouvé dans l'équipe pour suivre les résultats des tests dans la chaîne CI/CD.

On peut considérer l'ingénierie de la performance comme un nouveau mode de fonctionnement plus adapté aux dernières évolutions. Sa mise en place dans les projets permet de gagner en efficacité.
On n'a plus une équipe dédiée performance mais une prise en charge du sujet par l'ensemble des équipes et des acteurs du projet.

Les facteurs clés pour évoluer vers l'ingénierie de la performance

La performance n'est plus une activité ponctuelle réalisée juste avant la mise en production mais s'inscrit dans un continuum de la conception à la mise en production du service.
Au-delà des évolutions techniques, il s'agit bien d'un changement de « mindset », un élément à intégrer dès le début par l'ensemble des acteurs du projet.

2- Un mouvement contraint par des éléments contextuels

Dans une approche classique, les tests de performance ont lieu en fin du cycle de vie du projet, lors des phases d'intégration/qualification et juste avant la mise en production. Il est alors trop tard, notamment dans les cas où les tests de performance remontent des problèmes qui nécessitent de nouveaux développements ou une remise en cause de la conception, des choix techniques, de l'architecture. Fréquemment, les modifications/corrections retardent la mise en production et engendrent des coûts supplémentaires. Les coûts liés à la correction des problèmes sont d'autant plus importants que ces problèmes sont remontés tardivement. Il apparaît comme fondamental de tester le plus tôt possible et ceci dès les premières phases du cycle de vie du projet.

Un autre élément est lié au processus agile et à l'impératif de déployer le plus tôt possible l'application en production. Les tests de performance sont à réaliser à chaque sprint. On passe alors sur une approche itérative avec des tests à réaliser en continu.

Une approche comparative est utilisée pour évaluer les tests et les vérifier. On vient comparer les performances de la dernière version de l'application avec la précédente. Une dégradation

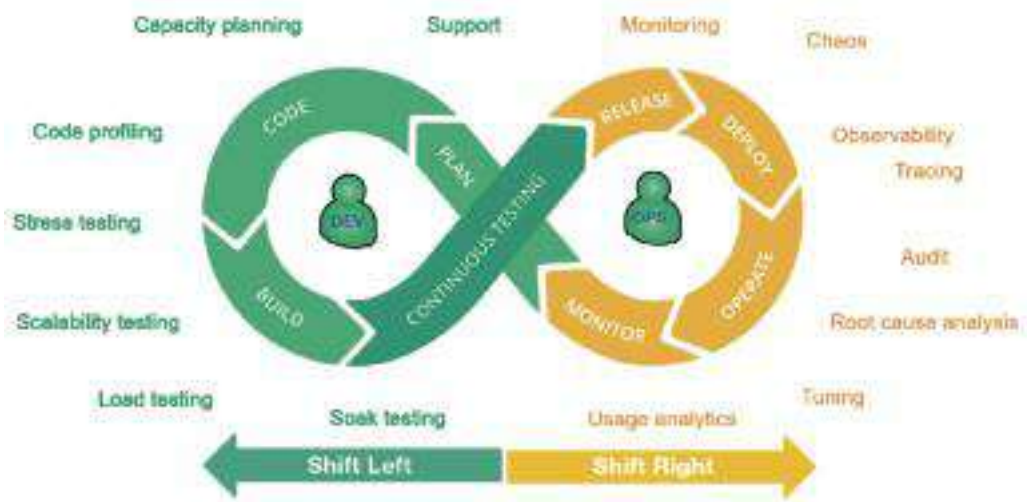
des performances indique une régression et une cause potentiellement liée aux dernières évolutions (i.e. modification du code).
Les tests ont vocation à être réalisés fréquemment et rapidement pour remonter le plus tôt possible les problèmes aux équipes de développement (« fast failed and fast feedback »).

3- Shift left, shift right et feedback loop : les étapes clés de l'ingénierie de la performance déclinées dans une démarche DevOps

Shift left : les tests de performance sont réalisés le plus tôt possible lors de la phase de développement et par des développeurs.
Les tests de performance sont intégrés dans une chaîne d'intégration et de déploiement en continu. L'automatisation permet de réaliser les tests rapidement et fréquemment.
Les outils de tests de charge ne sont plus utilisés par une équipe dédiée performance (i.e. un centre d'excellence performance) et des experts performance. Ils ont vocation à être utilisés par des développeurs. Les nouveaux outils privilégient une approche type développement, basée sur un ou plusieurs langages de programmation pour l'écriture des scripts et des scénarios de tests.

Shift Right : l'application déployée en production est surveillée (monitoring). Des indicateurs clés de performance sont définis et suivis via la mise en place de tableaux de bord. Des seuils associés aux indicateurs clés permettent de remonter des alertes.
Les tests de performance peuvent être réalisés sur des environnements proches de la production (par exemple une production cachée) avec une infrastructure représentative et fidèle à la production. On évite des biais dans les résultats des tests.
Des stratégies de déploiement permettent également de tester l'application en production avec des utilisateurs réels. On expose une nouvelle fonctionnalité à une partie des utilisateurs avant une généralisation à l'ensemble des utilisateurs (canary release).
Une optimisation de l'application est possible en production (tuning).

Feedback loop : la boucle de retour consiste en une remontée d'informations de la production vers le développement. Elle permet d'affiner les besoins non fonctionnels de performance (ce que l'on doit tester et comment le vérifier), le modèle d'usage (comment l'application est utilisée), le modèle de charge (le niveau d'utilisation en fonction du temps), les jeux de données, la configuration des bouchons…
Les données issues de la production fournissent des informations sur les utilisateurs finaux de l'application : les cas d'usage, les types de navigateurs, les accès réseaux, les temps de réflexion…
Au final, les informations remontées permettent d'améliorer les tests réalisés avant la mise en production, avec une meilleure représentativité (tests et scénarios plus réalistes).
L'ensemble des trois phases permet de mettre en place un cercle itératif vertueux pour améliorer les performances de l'application.



L'ingénierie de la performance et ses activités

On peut résumer les activités clés de l'ingénierie de la performance :

- Définition des exigences non fonctionnelles de performance. La définition des exigences doit être la plus précise possible pour décliner les tests à réaliser, les métriques à surveiller et les critères d'acceptabilité à mettre en place.
- Conception avec prise en compte de la performance.
- Développement avec prise en compte de la performance.
- Tests de performance dans une chaîne d'intégration et de déploiement continu.
- Tests en production.
- Suivi des performances de l'application en production et du comportement des utilisateurs finaux.
- Détection de problèmes, analyse, identification de la cause initiale, résolution.
- Boucle de retour avec des informations de la production pour affiner les tests à réaliser.

4- L'observabilité, l'élément indispensable

Pour s'inscrire dans un cercle itératif vertueux d'amélioration des performances de votre application, le besoin essentiel est de pouvoir surveiller l'application et de comprendre son fonctionnement interne. La mise en œuvre de l'observabilité est un prérequis absolu pour tirer parti des tests de performance que vous allez réaliser.

On peut définir l'observabilité de la façon suivante : dans la théorie du contrôle, l'observabilité est une mesure de la façon dont les états internes d'un système peuvent être déduits de la connaissance de ses sorties externes. La compréhension du fonctionnement interne de l'application est possible, en externalisant des données qui fournissent des indications sur son comportement (par exemple les logs).

L'observabilité peut être considérée comme un attribut de votre application, au même titre que la facilité d'utilisation, la disponibilité, la sécurité, etc. On peut donc transformer une application en une application observable et cette transformation repose sur l'instrumentation. Lorsque l'on va concevoir une application, il faut prendre en compte l'observabilité et prévoir une instrumentation. Une fois que l'application est observable, elle peut être surveillée (monitoring). L'objectif est de fournir les informations pertinentes permettant de résoudre le plus efficacement possible les problèmes : les limites et goulots d'étranglement dans le cas des tests de performance.

5- Une réponse à un écosystème de plus en plus complexe

L'observabilité répond à de nouveaux besoins, liés à l'accroissement de la complexité des applications et des environnements. Parmi les éléments contribuant à accroitre les difficultés de compréhension du comportement des applications, on peut citer :

- Le passage d'applications monolithiques vers des applications distribuées du type micro-service.
- La conteneurisation avec plus de difficultés pour accéder aux informations internes de l'application (problématique de boite noire).
- La multiplication des couches techniques : hardware, IaaS, CaaS et PaaS avec à chaque fois des couches d'abstraction supplémentaires.
- Une durée de vie éphémère des composants. Ils ont vocation à être instanciés et supprimés plus fréquemment en fonction des nouvelles versions ou des besoins de montée en charge (scaling).
- L'apparition de nouvelles architectures avec du multi cloud, du cloud hybride, de l'edge computing,…

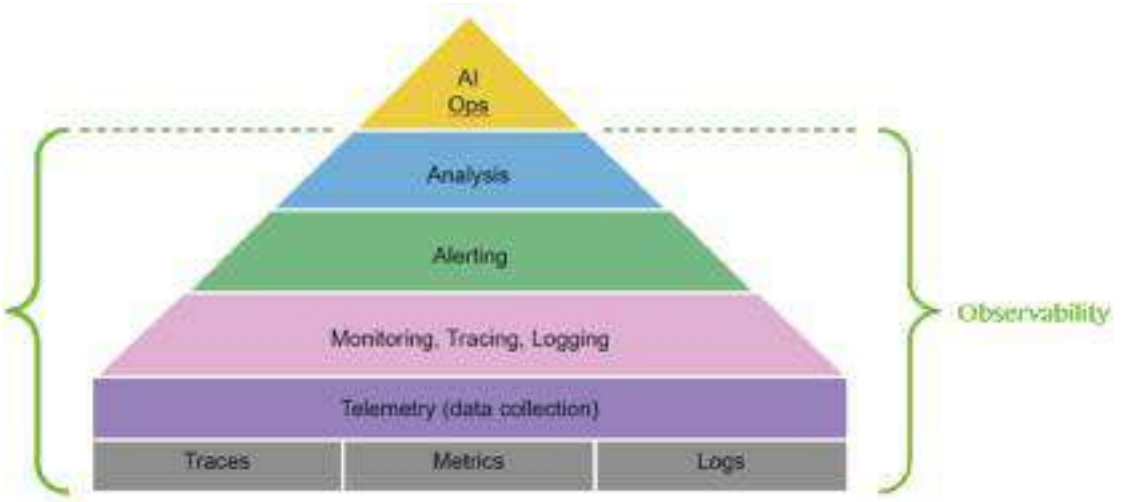
L'observabilité prend en compte ces éléments avec une approche basée sur le traitement et l'exploitation des données remontées par les applications.

6- L'instrumentation des applications

Quand on parle d'observabilité, on évoque souvent les trois piliers que sont les métriques, les logs et les traces applicatives (traces distribuées). Malgré cette approche par silos, l'objectif n'est pas de regarder chaque donnée de façon isolée mais au contraire de les corréler pour en tirer le maximum d'enseignements et rendre le système plus observable. L'observabilité permet alors d'avoir une vision globale, unifiée et de bout en bout.
L'observabilité repose sur l'instrumentation des applications pour recueillir les données exploitables. Un nouveau standard est devenu incontournable : OpenTelemetry. Il permet l'instrumentation, la collecte et l'export des données de télémétrie vers un back-end qui prend en charge notamment le stockage et la visualisation.

La norme offre une mise en œuvre indépendante des solutions propriétaires des fournisseurs et permet ainsi de changer facilement de back-end sans impacter les applications.

7- Une vision globale de l'observabilité



La pyramide de l'observabilité

La représentation de l'observabilité sous la forme d'une pyramide permet d'avoir une vision globale. On retrouve à la base les données (a minima les trois piliers), ensuite la couche de télémétrie qui peut être normalisée grâce à Opentelemetry. Les données sont traitées, stockées et visualisées avec des processus tels que le monitoring, le logging et le tracing.
L'alerting permet de détecter des anomalies et de remonter des alertes. L'analyse des anomalies est basée sur les données remontées. Il est nécessaire de corréler les informations entre elles pour trouver l'origine du problème (route cause analysis).
Enfin, une correction est apportée de façon manuelle ou de façon plus ou moins automatique. L'AIops, basée sur l'intelligence artificielle et le machine learning, permet une correction automatique ou auto-remédiation.

Conclusion

Nous avons vu ensemble les éléments contextuels, les concepts principaux et les facteurs clés pour se mettre en route vers l'ingénierie de la performance.
Pour avancer sur cette route, l'observabilité est une des étapes indispensables pour détecter les problèmes, les corriger et améliorer la disponibilité/fiabilité de vos applications.

II.6- Parcours multimodaux – Penser ses tests pour éviter « l'inflation » !

par Michael Granier

Si l'on souhaite répondre au maximum aux besoins des utilisateurs dans le développement d'applications, une seule « route » ou un unique mode d'accès est insuffisant !
La multimodalité ou multicanalité, voire omnicanalité, est un enjeu actuel fort mais qui a, depuis déjà longtemps, apporté son lot de complexité aux tâches de tests applicatifs.

Il est important de définir avant tout ce concept de « multimodalité », puis de comprendre la façon dont ce dernier a impacté les tests au fil des époques jusqu'à atteindre aujourd'hui une combinatoire difficile à mesurer.

Nous partagerons aussi autour de quelques bonnes pratiques d'outillages, afin de répondre le plus efficacement possible à ce besoin de « tout, tout le temps, partout » !

Multimoquoi ?

Dans le domaine du développement applicatif, la multimodalité est la possibilité d'accéder aux services d'une application par différents moyens, interfaces ou outils.

Un exemple simple est l'accès d'un site d'e-commerce via un navigateur Web, mais aussi une application Mobile ou en responsive sur une tablette.

Sans surprise, la multimodalité, ou multicanalité, entraîne quasi-systématiquement une multiplication des combinatoires qu'il faut gérer, afin de s'assurer de la qualité de son applicatif.
Et cette dernière n'est souvent testable que sur une interface plutôt stabilisée (pour les interfaces graphiques), donc à un stade avancé du produit.

Il reste cependant important de prendre en charge les exigences de multimodalité en amont du projet, pour veiller à ce que les utilisateurs puissent accéder qualitativement à votre application.
C'est ce que je vous propose de partager via un survol du test multicanalité à travers les époques !

Je vous parle d'un temps*…

La multimodalité reste un enjeu majeur et d'actualité mais, contrairement à ce que l'on pourrait penser, ce n'est pas une problématique récente dans le monde du test logiciel.

Vérifier que son application fonctionne avec différents points d'entrée, voire différentes configurations, existait avant l'ère de l'internet.
Déjà, à l'époque, les applications installées (appelées aujourd'hui « client lourd ») pouvaient être accessibles via différentes interfaces ou terminaux, souvent reliés par un fil ou en liaison radio locale.

* Sur un air de Charles Aznavour

Quelles configurations multicanales tester alors ?

Plus utilisés dans les domaines de la logistique ou de l'industrie, les tests multimodaux étaient cependant plus simples et ciblés en général sur une configuration spécifique de terminaux associés, avec peu de combinatoire. A notre époque, il faudrait s'imaginer que l'application fonctionne exclusivement sur un Windows dans une version définie avec un seul type de navigateur, et que toutes les autres combinaisons étaient considérées non prises en charge et donc non testables.

Peu de combinatoire, donc des tests multicanaux plus simples à cadrer et réaliser sur chaque interface définie.
Mais déjà à ce moment, la différence de fonctionnement de ces différents terminaux entraînait des exigences de validation multimodale à prendre en charge.

Et l'important pour les tests, dans ce cadre, est bien sûr de prendre en compte les quelques configurations demandées pour s'assurer que les applications fonctionnent toutes de manière optimale, en priorisant la configuration principale, mais aussi en s'assurant que les configurations précédentes, jusqu'à une certaine antériorité à définir dans la politique de test, soient toujours opérationnelles.

Il est venu le temps des interneeeeet*

Les années 90 marquent l'arrivée d'internet grand public, donc de la possibilité de proposer des « applications » sur la toile, et le début du commerce en ligne pour tous.

Même si l'explosion du e-commerce a attendu quelques années et que les entreprises concernées n'ont commencé à avoir réellement de stratégie de tests multicanaux qu'avec la multiplication des navigateurs, très souvent fournis par les opérateurs internet (rappelez-vous AOL, Club internet, Netscape, Firefox, opéra et bien sur internet explorer…), les acteurs du web ont vite constaté que les différents navigateurs utilisés par les internautes pour accéder à leurs sites pouvaient provoquer de nouvelles anomalies de compatibilité… et qu'ils allaient devoir intégrer cette nouvelle variable dans leur stratégie de test.

Il faut aussi penser à mentionner le WAP (Wireless Application Protocol), apparu en 1999 pour permettre aux « GSM » ou aux assistants personnels (PDA ou pocket PC, ancêtre des tablettes) d'accéder à une version quelque peu dégradée du net.
Même si celui-ci n'a pas connu le succès escompté, à cause d'un débit trop faible et l'obligation de créer des interfaces supplémentaires (et donc des coûts supplémentaires) pour être utilisés

*Sur un air de Notre-Dame de Paris

sur les téléphones de l'époque (parfois monochrome et avec une résolution faible), ces interfaces devaient faire partie des périmètres de tests multicanalités de toute application Web souhaitant être présente sur ce support… Mais vu le résultat, cela n'a pas dû être tout le temps le cas… !

Donc encore plus de canaux à tester ?

En effet, le fait de pouvoir accéder à un site web via un navigateur sur un type d'OS (windows, linux, MacOs…) constitue une forme de multicanalité importante, surtout quand des différences notables sont constatés entre ces différentes configurations.
N'oublions pas une particularité supplémentaire qui ajoutera à coup sûr du piment à la combinabilité : **la version du navigateur**. Certaines politiques d'entreprise, par exemple, ne permettent pas la mise à jour automatique des navigateurs et on se retrouve vite avec une version obsolète, mais sur laquelle l'applicatif doit absolument répondre aux différentes exigences… Ce qui entraîne forcément une gymnastique en termes de développement et de testing.

Rappelons un grand principe du test : les tests exhaustifs sont impossibles, et j'imagine souvent que ces problématiques de multicanalité étaient typiquement visées quand on a pensé à ce principe !

La donnée la plus importante à récupérer à ce moment est de connaître quel navigateur internet domine le « surf » !

Il faut ensuite axer ses tests sur les principaux navigateurs du marché.

Pour les versions, la priorité est souvent donnée aux 3 dernières versions maximum, les navigateurs se mettant automatiquement à jour dans une grande majorité des cas (sauf cas spécifique cité plus haut à prendre en compte dans sa stratégie).

Il ne faut pas non plus hésiter à faire évoluer son patrimoine en fonction des anomalies remontées, et axer ses tests sur le navigateur qui produit le plus d'anomalies (bonjour Internet Explorer).

En terme d'Operating System, la priorité sera là aussi donnée aux plus utilisés, c'est-à-dire Windows en priorité et MacOS par la suite. Il faut savoir que la version de l'OS n'a, en pratique, que peu d'impact sur la navigation internet et les bugs remontés, mais beaucoup plus sur les applications « lourdes ».

Et maintenant… que vais-je tester ?*

Avec l'avènement des Smartphones en 2007 (Iphone en tête), permettant d'accéder au Web via mobile en haut débit, mais aussi la mise en place du système d'app' mobile, qu'on retrouve maintenant en grand nombre sur iOS & Android, puis des tablettes en 2010, offrant la même capacité mais avec un plus grand écran, et enfin les « IoT » (Internet of Things) démocratisés avec les enceintes intelligentes connectées et la domotique, ces quinze dernières années ont vu l'explosion des moyens d'accès aux applications, donc, par essence, des combinatoires à tester…

En 2023, le trafic mobile a dépassé celui du Desktop (54%), il est donc absolument impossible de faire l'impasse sur les tests multicanalité d'un même applicatif.
Ces derniers peuvent maintenant être accessibles depuis des terminaux :

- Avec des écrans de taille différentes.
- Avec différents types d'OS et version d'OS (Android, iOS, Huawei…).
- Avec des couches constructeurs (Samsung, Huawei, Apple…) et opérateurs (SFR, Orange…) différents.

*Sur un air de Gilbert Bécaud

- Avec différents types de connexion (3G, 4G…).
- Avec un système d'App ou d'accès aux navigateurs avec du Web Responsive.
- Avec des usages différents selon le pays d'accès (internationalisation).
- Et parfois qui roulent ou volent...

Autant dire que l'exhaustivité est une utopie, voire une hérésie…

Il va falloir faire un choix !

Et un choix avisé et réfléchi : pour cela, **la data est votre meilleure amie.**

Métrique, analytics, tracking, logs… toute information concernant les utilisateurs qui parcourent votre applicatif sera un asset indispensable pour réaliser vos tests multimodaux sur les terminaux les plus pertinents.

Priorité au trafic : sans surprise, les terminaux les plus utilisés en termes de trafic seront à sécuriser en priorité. Plus il y aura d'utilisateurs à être impactés par un défaut ou une défaillance, plus la confiance et la e-réputation seront mises à mal. Idéalement, faire un TOP 10 (représentant souvent 80% du trafic) des accès à l'applicatif sera un minimum à prévoir.

Priorité au business : il faut à présent cibler les terminaux qui auront les meilleurs taux de transformation. Souvent, les modèles se recoupent et font partie du top précédent, mais ainsi recoupés, cela permet de donner une criticité plus importante aux tests sur ces devices.

Il est important de noter qu'il est déconseillé de prendre des données dites « généralistes » concernant les terminaux pour déterminer la configuration « idéale » à tester. Les données devront être prises dans le contexte de l'application à tester, car cette dernière n'aura pas la même cible, ni les mêmes utilisateurs. Et des disparités importantes sont souvent constatées d'un applicatif à un autre.

La priorité des terminaux est à présent établie, il faut maintenant définir quels tests omnicanalité mettre en place.

Là encore, connaître les usages de ses utilisateurs est primordial : savoir qu'ils consultent une information sur un mobile, qu'ils commandent sur un site Desktop, puis annulent sur un site responsive sur une tablette ; les parcours doivent être ciblés et ici aussi, un classement doit être établi pour déterminer ce qui doit être testé pour sécuriser 80% des usages.

Petit plus à mettre en place : il est assez rare dans un écosystème d'entreprise que tout le monde utilise le même terminal, que ce soit navigateur ou mobile. Proposer des phases de test exploratoire en « Crowd testing interne » où chacun va tester le ou les applicatifs avec sa configuration personnelle peut aider à remonter de nouvelles anomalies, en plus de faire découvrir le fonctionnel à tous.

« Parenthèse » sur le renforcement des tests de sécurité*…

Qui dit multiplication des accès à un applicatif dit multiplication des risques de failles de sécurité. Et ce ne sont pas les failles de sécurité importantes découvertes dans un nombre excessif

*Sur un air de rien du tout, je n'ai pas trouvé de musique à vous proposer ici

d'objets connectés, ayant entraîné quelques attaques ces dernières années, qui contrediront ce constat.

Sans rentrer dans le détail, nous vous invitons à bien penser aux tests de sécurité de vos terminaux et IoT dans votre stratégie de test multicanaux.

...et d'accessibilité

Les utilisateurs en situation de handicap ne représentent pas forcément le trafic le plus important ni même la plus grande valeur business, ils n'en restent pas moins des utilisateurs à ne pas ignorer, encore plus pour les sites d'état.
Il faudra donc, ici aussi, veiller à garder dans sa stratégie des tests avec des lecteurs de site Desktop & Mobile, afin de s'assurer que l'accessibilité soit respectée.

Chacun son device, chacun son automatisation*…

Première bonne nouvelle concernant les tests des multiples configurations de terminaux/navigateurs : un parcours Desktop, Web responsive ou App Mobile (iOS ou Android) devrait logiquement se comporter de manière quasiment identique d'une configuration à l'autre.

L'automatisation prend en compte des éléments du parcours qui ne dépendent pas du terminal ou du navigateur, sauf exception. La seule différence à prendre en compte, cependant, sera de prévoir un script par parcours : Desktop, responsive et app Mobile.
Vous pourrez donc en profiter pour tester votre « top navigateur » ou « top mobile » avec le même script automatisé de manière efficace sur différents terminaux et/ou navigateurs.

En ce qui concerne l'automatisation de parcours de bout en bout multimodaux, si vous cherchez à tout prix l'outil qui automatise tout en un seul parcours, vous allez vous arracher les cheveux. Dans l'idéal :

- Choisir un outil efficace pour automatiser les tests Web et répondant à vos besoins dans votre contexte projet (Responsive & Desktop).
- Un outil pour le Mobile ou autre type de terminal.

Pour optimiser l'automatisation sur Mobile, on aura souvent tendance à vouloir utiliser la même solution pour Android et iOS et des outils sauront faire cela avec plus ou moins de succès.
Il faudra cependant mettre en place des bonnes pratiques de développement pour avoir des parcours et éléments identiques sur les 2 App mobiles, sans quoi les différences vous obligeront à développer des scripts différents pour chaque OS Mobile, ce qui, par expérience, est souvent le cas. Mais ici aussi, le même script iOS ou Android pourra s'exécuter de la même manière sur le « **Top Devices** » de votre applicatif.

Il faut ensuite prévoir de « séquencer » les différents cas de tests automatisés de vos suites de tests les uns après les autres en récupérant les données de tests générés par le précédent :

*Sur un air de Tonton David

Desktop ou Mobile, pas d'importance puisque les tests ne sont pas liés par leurs outils d'automatisation mais par leurs données en entrée/sortie intégrées à leur exécution.

Pourquoi séquencer ? Afin d'éviter les scripts à rallonge qui valident une trop grande quantité de cas de tests.

Pour illustrer un exemple sur un site e-commerce :

- Création de compte Desktop.
- Activation par mail sur Mobile.
- Commande sur tablette.
- Après-vente sur Desktop.

Sauf si votre solution de test vous le permet (comme certaines solutions clés en main), il faudra prévoir une partie développement pour permettre ce séquencement.

Ainsi, vous pourrez simplement enchaîner vos parcours d'un canal à l'autre et simuler de vrais comportements utilisateurs multimodaux.

Quand on aura plus de tests, en l'an 2100* !

Êtes-vous prêts à tester l'interface de la prochaine voiture autonome volante ?
Il ne faut pas oublier qu'elle est pilotable via son téléphone et ses lunettes connectées ou son appareil de réalité virtuelle de dernière génération…
Et sur ce genre de véhicule, vous imaginez bien que les exigences de multicanalité sont plus que critiques, voire vitales !

On peut rêver d'une rationalisation des terminaux et d'une diminution de leur nombre au profit d'équipements moins nombreux et plus qualitatifs, mais ce n'est pas vraiment ce qui semble se profiler…
Alors, les tests multimodaux resteront un enjeu majeur, même dans le futur !

Mais pourquoi ne pas déjà imaginer des utilisations de l'intelligence artificielle pour des usages ciblés dans ce type de test, comme l'optimisation des configurations de tests à partir des données utilisateurs et pourquoi pas la génération automatique de scripts automatisés multisupport simplifiés ?

Dans tous les cas, l'exhaustivité des tests ne cessera de s'éloigner encore et encore, et avoir des aides efficaces pour faire le meilleur choix dans ses tests et éviter une inflation exponentielle va vite devenir indispensable.

*Sur un air de Pierre Bachelet

Sources :
- https://www.01net.com/actualites/le-wap-sur-la-touche-125722.html
- https://fr.wikipedia.org/wiki/%C3%89volution_de_l%27usage_des_navigateurs_web
- https://fr.wikipedia.org/wiki/Parts_de_march%C3%A9_des_navigateurs_web
- https://www.redhat.com/fr/topics/internet-of-things/what-is-iot
- https://www.linkedin.com/pulse/mobile-vs-desktop-sur-le-web-en-2023-georges-corre-expert-seo-sea/
- https://www.web-analytics.fr/formation/trafic-desktop-mobile
- https://www.lemondeinformatique.fr/actualites/lire-des-vulnerabilites-iot-donnent-un-acces-aux-reseaux-ot-89545.html
- https://www.cybermalveillance.gouv.fr/tous-nos-contenus/bonnes-pratiques/securite-objets-connectes-iot
- https://access42.net/developpement-web-quand-tester-avec-un-lecteur-d-ecran/

II.7- Le Chaos Engineering

par Stéphane Cazeaux

1- Pourquoi le Chaos Engineering ?

Par définition, les tests ont pour but de valider qu'un système se comporte tel qu'il a été conçu, donc de valider les comportements connus de ce système. La discipline du Chaos Engineering se distingue des tests par le fait qu'elle a pour but de chercher et comprendre des comportements inconnus du système, en injectant des stimuli qui permettent d'observer la réaction du système. Donc, contrairement aux tests, le Chaos Engineering ne s'appuie pas sur un comportement connu, mais sur une hypothèse.

Avant de rentrer plus en détail sur la définition de cette discipline, nous allons nous intéresser au pourquoi, et notamment comprendre pourquoi cette discipline devient importante aujourd'hui et à quel besoin elle répond.

Les systèmes que nous opérons sont de plus en plus complexes et nous les opérons dans un écosystème qui se complexifie également. Cette double complexité nécessite de repenser la manière dont nous concevons les systèmes et, en particulier, la manière de les tester.

Il n'existe pas une unique définition de la complexité, en revanche nous pouvons nous appuyer sur la notion de système complexe adaptatif. De manière basique, un système complexe adaptatif se définit comme un ensemble d'agents qui interagissent entre eux et forment un tout. Rapportés au logiciel, les agents peuvent se définir comme les composants logiciels (micro-services, composants applicatifs, …) mais aussi les acteurs (développeurs, testeurs, exploitants, clients, …). Même s'il n'existe pas de définition unique d'un système complexe adaptatif dans le cadre du logiciel, tout système complexe adaptatif présente des caractéristiques dont notamment l'irréductibilité. L'irréductibilité se définit par le fait qu'on ne peut dériver ou réduire un système sans perdre des propriétés significatives de ce système. Cette caractéristique a une grande importance dans le logiciel car elle implique qu'on ne peut considérer un système qu'en se focalisant sur une partie de ce système (ce qu'on fait généralement dans les tests), et qu'un retour arrière (après modification du système) n'est pas toujours possible sans altérer les propriétés du système.

Face à un système complexe, il devient extrêmement compliqué de connaître le fonctionnement du système a priori, et d'anticiper les impacts que peut avoir le moindre changement. Si cela reste faisable aux bornes d'un des agents du système, comment peut-on réaliser des tests visant à valider le bon fonctionnement d'un système complet ? Vient alors la question de la stratégie face à un système complexe.

Sur ce sujet, nous pouvons citer les travaux[1] d'Enrico Zaninotto et Kent Beck, qui datent de 2002.

[1] https://medium.com/@kentbeck_7670/cloud-development-environments-tame-complexity-by-reducing-state-4a154ea7959f

Enrico Zaninotto explique que la complexité peut être domptée en éliminant au moins un des quatre attributs suivants :

- **Dimensionnalité** : maîtriser la dimensionnalité, c'est le nombre d'états que peut avoir un système.
- **Interdépendances** : maîtriser les interdépendances entre les agents d'un système.
- **Incertitude** : maîtriser les changements venant de l'extérieur d'un système (évolution du marché, de la réglementation, des fournisseurs, …).
- **Irréversibilité** : à voir comme la capacité à ce qu'un design ne soit pas irréversible, autrement dit que la temporalité et les liens entre les décisions ne soient plus un élément critique, car un système pourra s'adapter. Réduire l'irréversibilité signifie donc qu'on peut accepter de nouveaux signaux auxquels on peut s'adapter.

Dans le logiciel, le seul attribut que l'on peut éliminer est l'irréversibilité. En effet, par nature, le logiciel encourage la prolifération d'états, donc éliminer l'attribut de dimensionnalité reviendrait à réduire les fonctionnalités du logiciel. Le logiciel encourage également à l'utilisation de librairies et API et donc à la création d'interdépendances. Enfin, l'écosystème dans lequel nous créons les logiciels est par nature incertain (aucune entreprise du logiciel ne peut maîtriser le marché, ni l'ensemble de l'écosystème logiciel).

En quoi consiste la suppression de l'irréversibilité ? Contrairement à ce que l'on pourrait penser de prime abord, supprimer l'irréversibilité ne se limite pas à permettre à ce que tout changement logiciel soit réversible (dont une implémentation concrète est le retour arrière), du fait qu'un système complexe est irréductible. La réversibilité va être apportée par la capacité que le système aura à s'adapter rapidement à des changements. Derrière cette notion d'adaptation rapide se trouvent notamment la capacité de déployer rapidement et la capacité de prendre des décisions rapidement.

Le Chaos Engineering a trouvé ses origines dans ce contexte de complexité avec plusieurs objectifs :

- Compenser par la pratique du Chaos Engineering ce qui ne peut plus être réalisé en test classique.
- Accompagner la nécessité de déployer rapidement (dans un but de réversibilité) en permettant de vérifier en continu la fiabilité du système.
- Accepter la panne comme un comportement normal d'un système complexe, et entraîner la réponse aux incidents pour augmenter la fiabilité (un système sera perçu comme stable si les incidents sont résolus rapidement voire automatiquement).

2- L'origine du Chaos Engineering

2.1- L'histoire de Netflix et Google

L'histoire la plus connue est l'origine du Chaos Engineering chez Netflix en 2008, avec notamment la création du premier outil Chaos Monkey. Mais nous allons nous intéresser à deux autres histoires moins connues.

La première, qui concerne toujours Netflix, est racontée par Nora Jones et Casey Rosenthal dans leur livre publié chez O'Reilly, où ils expliquent notamment que la pratique du Chaos Engineering a en quelque sorte était découverte par inadvertance. L'élément déclencheur a été la migration des services Netflix de leur datacenter in-house vers Amazon Web Services. Après cette migration, qui a eu lieu en 2008, Netflix a subi une mauvaise qualité de service due aux perturbations inhérentes à AWS (et au principe général du cloud), auxquelles les développeurs n'étaient pas préparés. Pour Netflix, il a donc fallu faire prendre conscience du problème à résoudre, ce qui est passé par la provocation volontaire des pannes (via l'outil Chaos Monkey) et un focus donné aux équipes sur le fait que, tant que ces pannes provoquaient des baisses de qualité, leur seule et unique priorité était de les résoudre. Un des éléments clés de la culture Netflix[2] est « highly aligned, loosely coupled », ce qui signifie qu'ils ont fortement aligné les développeurs sur le problème à résoudre, sans prescrire la manière de le résoudre (chaque équipe était libre de faire ce qu'un ingénieur sait faire le mieux : résoudre des problèmes). En cela, Netflix dit que le Chaos Engineering a été pour eux l'implémentation d'un principe de management, sous forme de code. Cette pratique ayant porté ses fruits, elle a ensuite évolué pour être le Chaos Engineering dont nous parlons aujourd'hui.

La seconde concerne Google qui, dès 2006, a initié un programme du nom de DiRT (Disaster Resiliency Testing). Le programme DiRT[3] repose sur la conviction que seul l'environnement de production est représentatif des conditions réelles et se focalise donc à la fois sur « tester la production » et « tester en production ». Le but du programme DiRT est de s'assurer que le système ne présente pas de faiblesse inconnue, mais aussi d'entraîner en permanence les équipes dans la gestion opérationnelle du service et de vérifier que les procédures sont toujours à jour.

2.2- Le Chaos Engineering comme évolution nécessaire des pratiques autour du logiciel

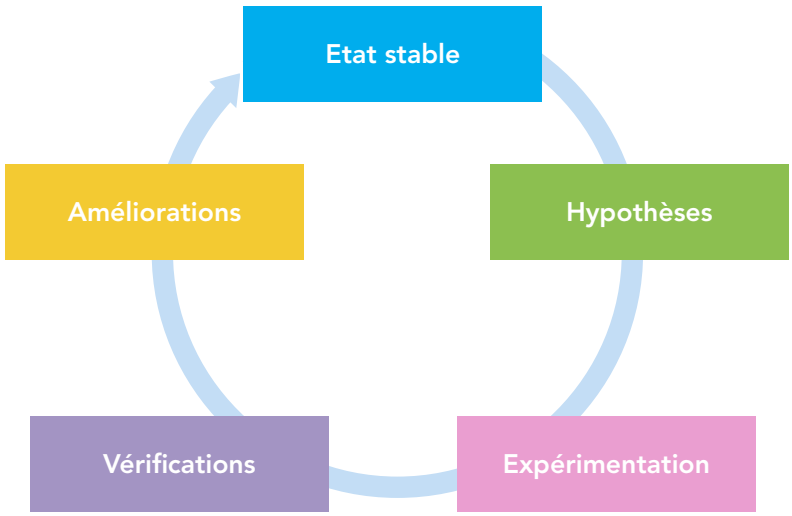
Le Chaos Engineering est une discipline déjà pratiquée depuis longtemps dans certains domaines comme l'aéronautique ou l'aérospatiale. Dans ces domaines, l'expérimentation dans le but de comprendre le fonctionnement d'un système (et d'en trouver des comportements ou faiblesses inconnus) fait partie des pratiques courantes. Donc, en tant que tel, le Chaos Engineering n'est pas fondamentalement nouveau.
Ce qui est nouveau, en revanche, c'est le fait de le pratiquer dans le domaine du logiciel et plus globalement de l'IT. En effet, avant l'arrivée des nouvelles pratiques autour de l'Agilité (et du Chaos Engineering dont nous parlons), le logiciel dans l'IT suivait les pratiques du cycle en V avec notamment une gestion des problèmes par des processus de validation et gestion de risque en amont. Cette stratégie focalise les efforts sur l'amont dans le but d'éviter les problèmes en production.

[2] https://jobs.netflix.com/culture
[3] https://queue.acm.org/detail.cfm?id=2371516

Avec la complexification croissante des systèmes, cette gestion en amont devient beaucoup trop compliquée, car plus personne ne devient capable de prédire les impacts qu'un changement peut induire aux bornes d'un système complexe. En effet, dans un système composé de milliers de micro-services, pour lequel il est impossible de reproduire la configuration de production dans un environnement de test, comment peut-on s'assurer qu'un changement ne va pas induire une dégradation ?
Comme nous l'avons vu en introduction, supprimer l'irréversibilité est le facteur clé de gestion de systèmes complexes dans le logiciel, et la vélocité (de déploiement et d'adaptation) est le moyen nécessaire. Dans ce cadre, le Chaos Engineering vient compléter ce qu'il n'est plus possible de faire dans les cycles de test et validation en amont. C'est bien ce qui explique l'intuition initiale de Google et Netflix dans leur démarche initiale du Chaos Engineering.

3- Les principes de base

Le Chaos Engineering s'appuie principalement sur le principe d'expérimentation venant de la méthode scientifique. Le diagramme ci-dessous illustre les étapes d'une expérimentation :



Dans la pratique du Chaos Engineering, chacune de ces étapes est importante :

- **Etat stable** : toute expérimentation démarre par la connaissance de l'état stable du système. Cette étape n'est pas anodine, en particulier lors des premières expérimentations, car le simple fait de formaliser l'état stable et de se mettre en capacité de le mesurer objectivement (via des métriques par exemple) permet d'en apprendre déjà beaucoup sur le système. Il s'agit donc ici de définir notre état stable et la manière de le mesurer.
- **Hypothèses** : les hypothèses définissent, selon notre connaissance du système, les impacts qu'aura l'expérimentation sur l'état stable.
- **Expérimentation** : il s'agit de l'étape de réalisation de l'expérimentation. En préparation de cette étape, il faut évidemment définir techniquement la façon dont l'expérimentation sera exécutée (quelle faute injecter, via quel outil), mais aussi définir la manière de

contrôler le rayon d'action de la faute (certains composants, certains utilisateurs, …) ainsi que les procédures de retour arrière en cas de problème grave.

- **Vérifications** : durant cette étape, les hypothèses seront confrontées au comportement réellement observé. Cette étape doit également servir à chercher des éléments encore inconnus (comportement du système, faiblesses, …) au-delà de la simple vérification des hypothèses.
- **Améliorations** : l'expérimentation permettant de créer de la connaissance, cette étape va permettre de capitaliser sur la connaissance acquise. Soit par amélioration du système (si une faiblesse a été détectée), soit par amélioration de la documentation (si un nouveau comportement a été découvert et compris).

Le Chaos Engineering ne se limite pas à l'étape d'expérimentation. Toutes ces étapes sont très importantes dans le déroulé d'une expérimentation car c'est le suivi rigoureux de ces étapes qui permettra de créer de la valeur à l'issue de l'expérimentation, cette valeur étant principalement liée à l'amélioration de la connaissance du système. Pour rappel, le Chaos Engineering trouve toute sa valeur dans les systèmes complexes où le système nécessite des expérimentations pour en comprendre le comportement.

4- Comment le mettre en pratique ?

4.1- L'acculturation

La mise en pratique doit d'abord débuter par l'acculturation des équipes. Et cela concerne tous les métiers de l'équipe : architecte, développeur, testeur, intégrateur, responsable produit, scrum master, etc.

L'acculturation permet d'apporter une définition claire de la pratique. En effet, le Chaos Engineering est souvent perçu comme un principe d'injection de faute, ce qui n'est qu'une toute petite partie (et la partie la plus technique) de la discipline. Durant l'acculturation, il est important d'insister sur le fait que le Chaos Engineering s'adresse au système dans son ensemble et vise à promouvoir la vérification, plutôt que la validation, dans la gestion d'un système complexe qui, a priori, ne peut pas être intégralement compris.

Il est également important de souligner que le Chaos Engineering ne se limite pas à rechercher les faiblesses logicielles, mais aussi les faiblesses liées à toute la chaîne entrant en jeu dans la gestion d'incident (procédures, relation entre équipes, communication, …).

4.2- Le point de départ

La principale recommandation est de démarrer petit. Si les premières expérimentations peuvent être relativement simples et identifier des petites faiblesses, elles permettront de manière facile de prendre des décisions actionnables et de se familiariser avec la pratique, tout en voyant rapidement et concrètement des bénéfices.

Les premières expérimentations peuvent également être faites entièrement à la main, en mobilisant plusieurs métiers dans l'équipe, de sorte à ne pas être freiné par une mise en place initiale d'outils de Chaos Engineering. Les outils trouveront leur valeur dans la nécessité de rendre les expérimentations répétables (ou plus complexes).

Enfin, lorsque vient le choix d'expérimentation à réaliser, en particulier des fautes à injecter, la recommandation est de privilégier les fautes de type **latence**. En effet, en ralentissant une application (un composant, un micro-service), on simule le symptôme de la très grande majorité des pannes : un problème de ressources (CPU, mémoire), un problème de performance, une exception dans l'application, un ralentissement réseau, une connexion réseau défaillante (qui peut se simuler avec latence très élevée, de sorte à déclencher des timeout), etc. Si on est capable de ralentir une application, on est en mesure d'expérimenter la réponse d'un système à tout type de panne.

4.3- Les rôles

La question qui va se poser rapidement est le rôle d'ingénieur du chaos. Doit-on créer un nouveau métier ? Une équipe d'ingénieurs du chaos ?

Dans le principe, l'ingénieur du chaos doit être une personne qui connaît parfaitement bien le système (architecture, fonctionnement, chaîne de gestion d'incident, …) et est donc à même de savoir piloter des expérimentations. Par conséquent, l'ingénieur du chaos ne peut pas être une personne extérieure à l'équipe, ni un nouvel arrivant dans l'équipe. Ce rôle doit être porté par une personne ayant un bon niveau de connaissance du logiciel. Pour reprendre une citation de Nora Jones : *le Chaos Engineering n'est pas une tâche confiée à un stagiaire*.

Qu'en est-il de la possibilité de monter une équipe de Chaos Engineering en transverse ? Pour ce qui est de piloter et gérer les expérimentations, cette approche peut poser des problèmes : montée à l'échelle, connaissance fine du système, etc. Mais une telle équipe peut faire sens pour ce qui est de l'accompagnement initial et de l'acculturation.

Enfin, l'ingénieur du chaos doit être dans une position qui lui permet de piloter les expérimentations et de capitaliser sur les connaissances qui en découlent. Mais il ne doit pas être la personne qui priorise et suit les correctifs/améliorations du logiciel.

5- Les mises en application possibles

Il est possible de mettre en application le Chaos Engineering de plusieurs façons, non exclusives entre elles.

La plus immédiate est l'usage des outils d'injection de faute (voir paragraphe suivant) dans le cadre des tests (tests fonctionnels, tests de performance, etc.) dans le but de tester la fiabilité. Bien que tout à fait pertinente, cette approche se distingue assez peu du test classique, car elle amène à tester qu'une implémentation (liée à de la fiabilité) se comporte comme spécifiée.

Or, le Chaos Engineering a surtout pour but de chercher des comportements inconnus, que ce soit dans la recherche de faiblesse ou dans la nécessité de vérifier un comportement, qu'un système trop complexe empêche a priori de définir.

Nous allons nous intéresser à quatre solutions représentatives des mises en application possibles.

La première, qui est celle par laquelle il est recommandé de démarrer, est **l'expérimentation ad-hoc**. Il s'agit de mettre en place une expérimentation dans un contexte bien particulier : démarrage du Chaos Engineering avec une expérimentation simple, levée d'un doute ou d'une inquiétude sur un fonctionnement particulier, préparation d'une situation particulière (événement nécessitant de vérifier la résilience, incident passé, …). Dans tous les cas, le principe consiste à dérouler une expérimentation en suivant la méthode définie plus haut, avec les acteurs du produit concernés (de l'architecte jusqu'au déploiement) et de tirer parti de la connaissance qui sera acquise. Cette expérimentation peut être réalisée une unique fois, ou bien conservée pour devenir une expérimentation répétée.

Le **Chaos Day** (ou Game Day) se déroule sur une journée (ou demi-journée) et a pour objectif d'améliorer la fiabilité d'un système sous forme de jeu. Améliorer le système peut prendre plusieurs définitions : améliorer la réponse aux incidents, trouver des nouvelles faiblesses du système, entraîner les équipes opérationnelles, augmenter la prise de conscience de l'importance opérationnelle pour des équipes habituellement éloignées du terrain opérationnel, etc.

Le **Chaos Day** doit être préparé à l'avance, avec des règles du jeu et des rôles bien définis (ceux qui provoquent les fautes, ceux qui cherchent les pannes, les observateurs, etc.). L'utilisation de la forme du jeu a pour but de faciliter l'apprentissage et l'implication. Il a été observé, par exemple, que des équipes opérationnelles préfèrent monter en compétence sur un système à opérer de cette manière (en apprenant par le jeu comment répondre à des pannes), plutôt que par des formations traditionnelles.

La **vérification continue** a pour but de capitaliser sur des expérimentations passées et de les répéter (de manière automatique) dans le but d'identifier des changements. Le terme « vérification » s'oppose ici à « validation », dans le sens où le but est d'être capable d'identifier qu'un changement dégrade la fiabilité du système a posteriori (vérification), sur un système trop complexe pour qu'il puisse être vu a priori (validation). La mise en œuvre de la vérification continue nécessite des outils (orchestrateurs), décrits dans le paragraphe suivant. En raison de l'aspect automatique,

ces expérimentations vont s'appliquer sur l'aspect technique du système, contrairement à des expérimentations ad-hoc ou en *game day* qui peuvent s'intéresser aux aspects process et organisation.

Dans tous ces cas, sur quel environnement ces expérimentations doivent-elles être réalisées ? Idéalement en production, car c'est uniquement là que nous sommes confrontés aux conditions réelles du service, difficiles voire impossibles à reproduire en environnement de tests sur les systèmes complexes. Mais réaliser des expérimentations en production nécessite une bonne maturité sur la fiabilité du système et la pratique du Chaos Engineering.

Il est donc tout à fait pertinent de mettre au point les expérimentations dans un environnement hors production, voire de ne mener certaines expérimentations qu'en environnement hors production. Mais les réaliser en production doit être la cible à atteindre (à défaut d'être la première étape), en particulier pour la vérification continue qui n'a de sens qu'en production.

6- Les principaux outils

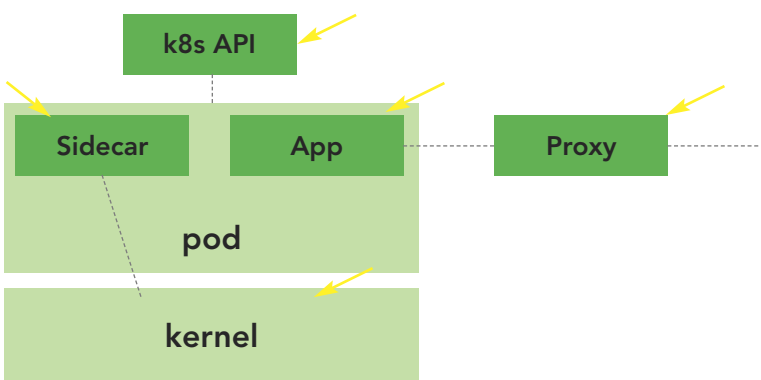
Nous allons à présent nous intéresser aux outils qui accompagnent la pratique du Chaos Engineering.

Avant de rentrer dans les détails, il est important de rappeler que le Chaos Engineering ne se limite pas à la mise en œuvre d'un outil. En effet, l'outil va faciliter la phase d'exécution de l'expérimentation, mais aucun outil ne permet de réaliser les phases de définition de l'état stable, des hypothèses et des vérifications.

De plus, un outil va nécessairement travailler sur la base d'une connaissance acquise (en particulier si l'outil effectue des vérifications). De ce fait, l'outil ne peut pas remplacer la phase de création de connaissance par l'expérimentation.

6.1- Injection de faute

L'injection de faute est l'élément de base d'une expérimentation. Le schéma ci-dessous représente les différentes possibilités d'injection de fautes disponibles actuellement (ce schéma s'appuie sur une architecture kubernetes, qui est le cas le plus complet) :



L'injection de faute au niveau du **noyau Linux** est rendue possible par la technologie eBPF, qui permet d'interagir de manière programmatique avec la pile réseau du noyau. Cette technologie offre de très nombreuses possibilités, dont notamment la simulation des perturbations au niveau du réseau. Il est possible d'utiliser eBPF directement (ce qui requiert des compétences de programmation avancées), ou de tirer parti d'outils qui exploitent eBPF, notamment parmi les services mesh comme Istio ou Cilium.

L'injection de faute la plus communément utilisée passe par l'utilisation d'un **sidecar**. Via ce principe, il devient possible d'agir au niveau du système pour perturber l'application qui s'exécute sur ce système. Typiquement, il va être possible d'exécuter des scripts qui vont surconsommer des ressources (mémoire, CPU, entrée/sortie disque) ou agir au niveau réseau (via iptables ou tc). Dans un environnement VM, on utilisera des scripts directement au niveau de la VM.

L'injection de faute au **niveau de l'application** est celle qui offre les perspectives les plus intéressantes. Elle consiste à implémenter des comportements fautifs comme fonctionnalités directement dans l'application, et de les rendre pilotables via une API. Un principe de non-persistance (configuration remise à zéro et injection de faute désactivée au démarrage de l'application) afin de sécuriser l'usage de cette fonctionnalité en environnement de production. Cette approche présente de très nombreux avantages :

- Elle permet d'injecter les fautes directement dans l'application, avec possibilité de cibler des éléments particuliers de l'application, comme l'exécution d'une logique particulière (pour générer une exception ou de la latence), la connexion à une base de données, à une API. Contrairement aux autres approches où les fautes sont injectées par l'extérieur, ce qui limite la finesse de ce qu'il est possible de faire.
- Du fait d'implémenter les fautes possibles, elle permet une meilleure prise de conscience et implication des équipes qui construisent le logiciel (architectes, développeurs, testeurs, …).
- Elle rend l'injection beaucoup plus simple et indépendante du déploiement de l'application. En effet, les autres approches peuvent parfois être complexes à mettre en œuvre, alors que l'injection de faute au niveau de l'application est aussi simple à utiliser que toute fonctionnalité de l'application.

L'injection de faute au niveau de l'application est actuellement encore émergente et peu de librairies sont disponibles. La plus notable actuellement est **Chaos Monkey for Spring Boot**[4] qui, comme son nom l'indique, apporte ces fonctionnalités d'injection de faute (latence, exception, consommation CPU/mémoire) pour des applications basées sur Java Spring Boot. Il n'existe pas d'autre librairie notable pour d'autres environnements.

L'injection de faute au niveau de l'API Kubernetes (ou plus généralement, de l'API de l'infrastructure cloud) consiste à utiliser les API de gestion de l'infrastructure pour simuler des fautes.

[4] https://codecentric.github.io/chaos-monkey-spring-boot/

L'exemple le plus typique consiste à tuer un pod (c'est ce que faisait le premier outil de Netflix, Chaos Monkey). D'autres fautes sont possibles, comme restreindre les ressources ou modifier les flux réseau.

Enfin, l'injection de faute au niveau du proxy permet de simuler notamment des perturbations réseau entre des composants. C'est utile lorsqu'on souhaite, par exemple, simuler de la latence d'un composant externe qu'on ne contrôle pas. Plusieurs proxys offrent ces possibilités d'injection de fautes, dont notamment toxiproxy et envoy.

6.2- Orchestrateurs

Dans la pratique du Chaos Engineering, l'usage des outils d'injection de faute se fait via des orchestrateurs, qui permettent de réaliser une expérimentation complète :

- Vérifier l'état stable au début.
- Sélectionner les cibles (par exemple en utilisant les API kubernetes) sur lesquelles les fautes sont injectées.
- Injecter une ou plusieurs fautes, en parallèle ou en séquence.
- Vérifier les hypothèses tout au long de l'expérimentation.
- Faire le retour arrière en fin d'expérimentation ou en cas d'interruption.

Parmi les orchestrateurs notables nous avons :

- ChaosToolkit : implémenté en Python, il a la particularité d'être agnostique à l'infrastructure cloud et peut donc très facilement se mettre en place pour démarrer des expérimentations. En revanche, cet outil se montre assez limité pour des expérimentations complexes et dans ses possibilités de vérification.
- LitmusChaos : implémenté en Golang, il a la particularité d'être adhérent à Kubernetes. Il doit être déployé sur un cluster Kubernetes et ne peut injecter des fautes que sur des applications s'exécutant dans un cluster Kubernetes. Contrairement à ChaosToolkit, il est donc plus complexe à mettre en œuvre, mais il offre également beaucoup plus de possibilités et une capacité de vérification très poussée.

Il est important de noter que ces orchestrateurs offrent des capacités de vérification (vérification de l'état stable, vérification des hypothèses), ce qui implique qu'il doit être possible d'implémenter ces vérifications. Soit via des probes classiques, soit via des métriques (avec Prometheus par exemple). Par conséquent, ils ne peuvent réaliser que des vérifications déjà connues. Ces outils ne peuvent donc pas remplacer des vérifications manuelles pour chercher de l'inconnu.

6.3- Autres outils

Comment nous l'avons plusieurs fois souligné, le Chaos Engineering vise à créer de la connaissance. Parmi les outils nécessaires à la pratique, il est donc important d'avoir un outil permettant de tracer et capitaliser sur ce qu'on doit apprendre d'une expérimentation.
Il n'existe aucun outil Chaos Engineering qui offre cette possibilité, mais des outils classiques comme un wiki ou Confluence sont parfaitement adaptés à cela.


ON A DES PROBLÈMES AVEC LES IA GÉNÉRATIVES QU'ON A MISES À L'ÉCRITURE PUIS À LA CORRECTION DU CODE...
ELLES HALLUCINENT?
NON... ELLES SE CRÊPENT LE CHIGNON.
IMBÉCILE! TU METS TROP DE BUGS DANS TON CODE!
C'EST TOI QUI NE LES CORRIGE PAS ASSEZ VITE, TÊTE DE MÛLE!
BONG BONG
AUTO TEST
AUTO CODE
IAKEMOIKITESTE
IAKEMOIKICODE
F. COINTE

PARTIE III
L'IA POUR LES TESTS ET TESTS DE L'IA

III.0- L'IA pour tester est là - Apprenons à travailler avec dès maintenant

par Bruno Legeard

Cela n'a échappé à personne : l'IA et particulièrement l'IA générative est arrivée dans notre domaine des tests logiciels : dans la communication mais aussi dans les plans stratégiques de nos entreprises.

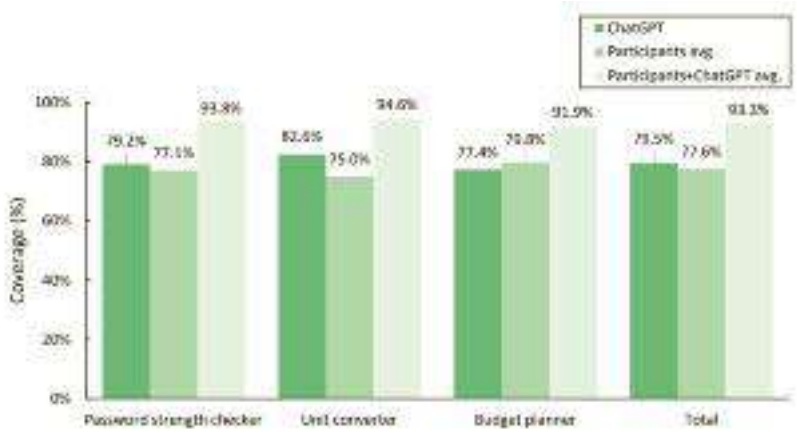
Pourquoi cet engouement ? Est-ce une nouvelle illusion technologique ? Est-ce un feu de paille ?

Il est bien sûr difficile de répondre à ces questions de façon définitive aujourd'hui. Mais force est de constater que les grands modèles d'IA générative (voir le chapitre suivant pour plus de détail), qu'ils soient commerciaux ou utilisables en licence open-source, obtiennent d'ores et déjà des résultats impressionnants sur des tâches de test.

Vous allez peut-être me dire que les résultats avec une IA générative, par exemple en conception de tests ou écriture du code d'automatisation, sont loin d'être parfaits. Mais n'est-ce pas aussi le cas pour nous, humains ? L'IA générative peut être approximative, incomplète ou peu précise, et parfois excellente et proposer une réalisation complète de la tâche, comme nous donc !

La synergie Humain et IA générative pour les tâches de test

La figure suivante est extraite d'un article de Janvier 2024 intitulé : « ChatGPT and Human Synergy in Black-Box Testing : A Comparative Analysis »[1]



Cet article, écrit par des chercheurs de NTT au Japon, explore les capacités de ChatGPT (GPT-4) à générer des cas de test en boîte noire par rapport aux testeurs humains. L'expérience réalisée met en parallèle la conception des cas de test en mode boîte noire pour trois applications, d'une part par ChatGPT, d'autre part par quatre testeurs humains.

[1] Cf. https://arxiv.org/abs/2401.13924

Le graphique montre clairement que sur cette expérience et en analysant la couverture obtenue, par rapport à une complétude définie par les auteurs :

1. L'IA surpasse parfois les humains (en moyenne de la couverture obtenue).
2. L'IA est parfois sujette à erreur et confusion.
3. Mais, encore plus important, la couverture combinée Humain & IA permet d'atteindre un excellent score sur chacune des trois applications testées (> 90% de couverture).

Cette expérience est bien sûr limitée, ces trois applications sont un faible échantillon et cela demanderait à être reproduit. Mais n'est-ce pas une indication claire du chemin à prendre : intégrer les IA comme nos assistants de test en conservant notre responsabilité ? L'IA nous assiste au quotidien mais nous sommes les sachants et décideurs pour assurer la bonne réalisation de la tâche de test qui nous est confiée.

Maîtriser l'usage de l'IA générative pour les tests : il est urgent de se former

L'arrivée à maturité de l'IA générative et sa disponibilité pour accélérer les tâches de test, ouvrent de nouvelles perspectives mais aussi des défis et problématiques à considérer.

Le premier défi est d'en maîtriser l'usage, de définir les bonnes pratiques et de l'intégrer de façon optimisée dans nos processus de test. Pour cela, une seule réponse : il faut se former, chacun de nous, et que les organisations forment les équipes de test collectivement.

Comment imaginer pouvoir maîtriser cette technique sans acquérir les connaissances et nouvelles compétences que vont impliquer son usage ? Cela signifie en particulier :

- De connaître, par la pratique, les techniques d'utilisation de l'IA générative (requêtage, raisonnement, évaluation des résultats) pour les tests et d'acquérir cette nouvelle compétence.
- De connaître, en pratique, les risques et l'IA générative appliquée aux tests tels que les erreurs typiques de l'IA, les données de consommation d'énergie et les biais possibles dans les résultats, et de savoir se prémunir autant que possible de ces risques.
- De maîtriser l'intégration des applications de l'IA générative dans les processus de test pour en obtenir les résultats attendus.

Il existe de nombreuses ressources disponibles, y compris ce livre du CFTL et les tutoriels et conférences de la JFTL.

En conclusion, l'IA générative est là. Bonne nouvelle ou mauvaise nouvelle ?

Cela dépend en partie de nous. Comme toute innovation de rupture, nous avons des marges de manœuvre pour ne pas la subir mais au contraire pour bénéficier de la nouvelle donne. L'un des leviers est celui de la compétence technique, méthodologique ou organisationnelle sur l'IA générative pour les tests logiciels.

En développant nos compétences pour une intégration efficiente de l'IA générative et consciente des risques associés, nous favoriserons une maîtrise graduelle au service des équipes et des individus. Cette approche proactive de l'IA générative est certainement préférable à une position de crainte, de refus irréaliste et finalement d'un manque de maîtrise de la transformation qui arrive dans notre métier des tests logiciels.

III.1- Introduction à l'IA générative pour les tests logiciels

par Bruno Legeard

Les technologies d'Intelligence Artificielle (IA), dite IA générative, ont émergé de façon très large depuis 2022 avec la sortie de ChatGPT par la société OpenAI. En Novembre 2023, on comptait déjà plus de 100 Millions d'utilisateurs actifs par semaine sur ChatGPT, avec une poursuite continue de la croissance. La sortie de nouveaux modèles tels que Anthropic Claude 3 ou Google Gemini et l'intégration dans les moteurs de recherche accélèrent cette diffusion.

Au-delà de l'usage conversationnel (type ChatGPT), de très nombreuses applications en entreprises se développent. Cela concerne en particulier la création de contenu, le support au développement de produits, l'analyse marketing, financière, légale, l'amélioration de l'expérience utilisateur ou des applications dans le domaine de la santé. Cette accélération de l'usage de l'IA touche directement nos activités de l'Assurance Qualité (QA) et des tests logiciels : aide à l'analyse des anomalies, conception des tests, support à l'automatisation, etc.

Au cœur de ces applications se trouvent les grands modèles de langage pré-entraînés - appelés **LLM - Large Language Model**, et un écosystème complet de techniques, méthodes et outils pour offrir les services basés IA.

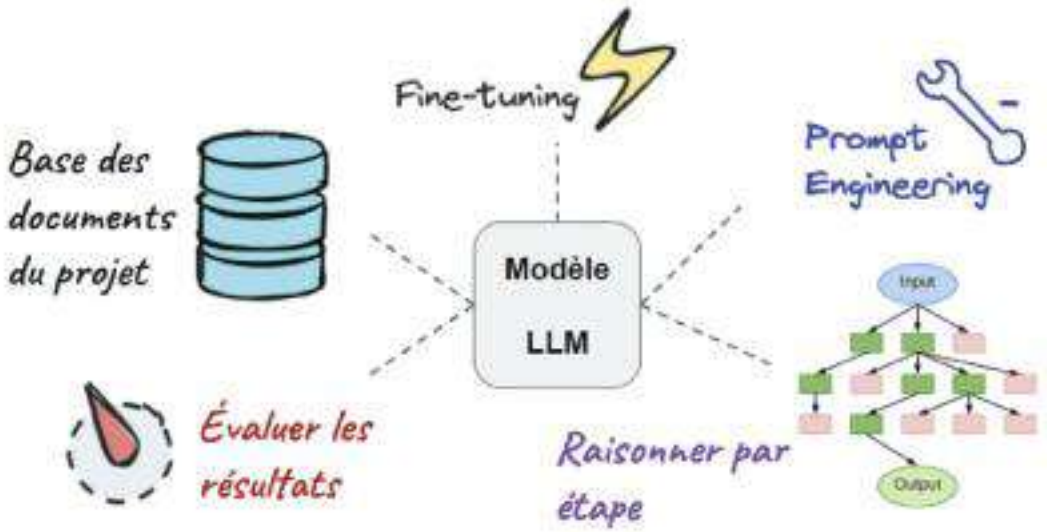


Figure 1 : L'écosystème de l'IA générative, au-delà du LLM

Ce chapitre du livre du CFTL vise à apporter les connaissances principales à mi-2024 sur les technologies d'IA générative/LLM et sur les techniques, méthodes et outils associés, avec une vue à 360° du domaine et de son application au domaine des tests logiciels. Les sujets clés des hallucinations des modèles, des risques de cybersécurité, d'éco-responsabilité et les questions de biais et d'éthique sont aussi abordés.

1- IA générative - de quoi parle-t-on ?

L'intelligence artificielle générative, ou IA générative, est un sous-domaine de l'IA (cf. figure 2) visant à générer du contenu à partir de modèle d'IA pré-entraîné. Contrairement à l'IA traditionnelle qui se concentre sur la reconnaissance et la classification des données, l'IA générative vise à produire de nouvelles données, telles que du texte, des images, de la musique ou même du code informatique, en s'appuyant sur des modèles appris à partir d'exemples existants.

Figure 2 – L'IA générative est un sous-domaine de l'IA focalisé sur la production de nouvelles données par des modèles d'IA pré-entraînés

L'histoire de l'IA générative remonte aux années 1950 avec les premiers travaux sur les réseaux de neurones artificiels. Mais ce n'est que récemment, grâce aux progrès de l'apprentissage profond (deep learning) et à la disponibilité de grandes quantités de données, que l'IA générative connaît un essor considérable. Les modèles génératifs actuels, tels que les réseaux antagonistes génératifs (GAN) et les modèles de langage comme GPT (Generative Pre-trained Transformer), ont ouvert de nouvelles perspectives dans de nombreux domaines d'application, nécessitant la production de nouvelles données.

Le différentiateur clé est le pré-entraînement du modèle d'IA générative sur de très grands corpus, tels que des textes, du code informatique ou des images. Ces modèles sont appelés LLM – Large Language Model, et ils sont créés en deux étapes (cf. Figure 3) :

1- L'entraînement d'un modèle **LLM de base**, entraîné sur d'énormes corpus de textes non étiquetés, capables de capturer les structures et les relations linguistiques complexes.

2- L'entraînement d'un **modèle LLM répondant aux instructions** par renforcement d'un modèle LLM de base, affiné pour suivre des instructions spécifiques et réaliser des tâches bien définies, comme la réponse à des questions, la traduction ou la génération de texte.

LLM de base
(Base LLM)

• Prédit le mot suivant à partir des données textuelles apprises et de la proximité linguistique

Il était une fois une princesse nommée Isabella. Elle vivait dans un grand château au sommet d'une colline verdoyante...

Quelle est la capitale de la Mongolie ?
Quelle est la superficie de la Mongolie ?
Quel est le nombre d'habitants de la Mongolie ?

LLM adapté aux instructions
(Instrution-Tuned LLM)

• Prédit le mot suivant à partir des données textuelles apprises et de la proximité linguistique
• Renforcé par des retours sur les réponses pour favoriser les bonnes réponses
• RLHF : Reinforcement Learning with Human Feedback

Quelle est la capitale de la Mongolie ?
La capitale de la Mongolie est Oulan-Bator

• Modération des réponses : éviter les réponses agressives, racistes, dangereuses,...

Figure 3 – Du LLM de base au LLM adapté aux instructions

Mise en garde :

Un aspect essentiel à toujours avoir à l'esprit lorsque l'on utilise un modèle d'IA générative, malgré l'effet bluffant d'une conversation avec ce modèle, est qu'il s'agit de probabilité et pas d'intelligence (au sens humain). Un LLM est construit pour générer le mot suivant d'un texte dans un contexte donné à partir d'un grand volume de données d'entrainement. Un modèle LLM produit du probable et le plus probable peut être faux. On dit que le modèle « hallucine ». Cette capacité d'erreur est constitutive des modèles d'IA générative. Chacun doit en avoir conscience et prendre garde aux fausses réponses du LLM.

2- Etat des technologies LLM

Le développement des modèles LLM s'est accéléré de façon très forte depuis fin 2022. On peut noter deux grandes tendances :

- Des modèles de plus en plus gros en taille de données d'entraînement et puissants en termes de capacité de réponses, tels que GPT-4 suivi prochainement par GPT-5 de OpenAI et Claude 3 de Anthropic, qui combinent en réalité plusieurs modèles LLM dédiés chacun à des activités/types de données. Cette course à la puissance des modèles LLM visent à exceller dans toutes sortes de tâches, de l'examen de mathématiques au baccalauréat à la génération automatique de scripts de tests automatisés (pour ne citer que deux exemples).
- Des modèles plus petits, avec un usage libre et gratuit, tels que Mistral 7B de Mistral et/ou CodeLlama de Meta, pouvant être adaptés à des tâches spécifiques (par exemple pour analyser et optimiser des cas de test) et mis en production sur une infrastructure de calcul au sein de l'entreprise utilisatrice.

Ces deux évolutions coexistent et permettent aux utilisateurs d'IA générative que nous sommes de pouvoir choisir ce qui nous convient le mieux, ou respecte le mieux les contraintes de notre organisation.

Dans les deux cas, un modèle de langage s'utilise via des requêtes appelés Prompt, le requêtage étant appelé le Prompting. Le Prompting est une technique essentielle pour interagir avec les LLM et obtenir les résultats souhaités. Il consiste à fournir un contexte et des instructions au modèle sous forme de texte, appelé prompt, afin de guider la génération de texte. Le prompt engineering est l'art de concevoir des prompts efficaces pour exploiter au mieux les capacités des LLM.

Parmi les techniques de Prompting courantes, on peut citer :

- **Few-shot prompting** : fournir quelques exemples dans le prompt pour guider le modèle vers la tâche souhaitée, sans nécessiter de réentraînement.
- **Chain-of-thought prompting** : inciter le modèle à décomposer un problème complexe en étapes intermédiaires explicites, améliorant ainsi la qualité et l'interprétabilité des réponses.

- **Zero-shot prompting** : décrire une tâche directement dans le prompt, sans fournir d'exemples, en s'appuyant sur les connaissances générales du modèle.
- **Prompting par rôle** : attribuer un rôle ou une personnalité spécifique au modèle dans le prompt, pour influencer le style et le contenu de la réponse.
- **Prompting par contexte** : inclure des informations contextuelles pertinentes dans le prompt, telles que des données utilisateur ou des connaissances spécifiques au domaine, pour personnaliser et affiner les résultats.

Le prompt engineering implique également des techniques pour optimiser la clarté, la spécificité et l'efficacité des prompts, telles que :

- La décomposition des tâches complexes en sous-tâches plus simples.
- L'utilisation d'un vocabulaire et d'une syntaxe appropriés pour le domaine et le public cible.
- L'itération et l'affinage des prompts basés sur les résultats obtenus.
- La gestion des prompts pour des tâches spécifiques ou des domaines d'application particuliers.

Un prompt engineering efficace permet d'exploiter pleinement le potentiel des LLM, en obtenant des résultats plus précis, cohérents et adaptés aux besoins spécifiques. C'est une compétence clé, un usage optimal de l'IA générative pour accélérer les activités de tests logiciels : sans un requêtage efficace du LLM, pas de gains possibles.

3- Les cas d'usage et les bonnes pratiques de l'IA générative pour les tests

La grande force de l'IA générative est la capacité des grands modèles pré-entraînés de découvrir des liens sémantiques entre informations textuelles, et d'utiliser cette capacité pour en créer de nouvelles, notamment. C'est très pertinent pour les tests. Par exemple, extraire les classes d'équivalence à partir d'une spécification métier ou générer des scripts BDD à partir de critères d'acceptation, exploite ces capacités d'analyse sémantique et de génération des modèles LLM.

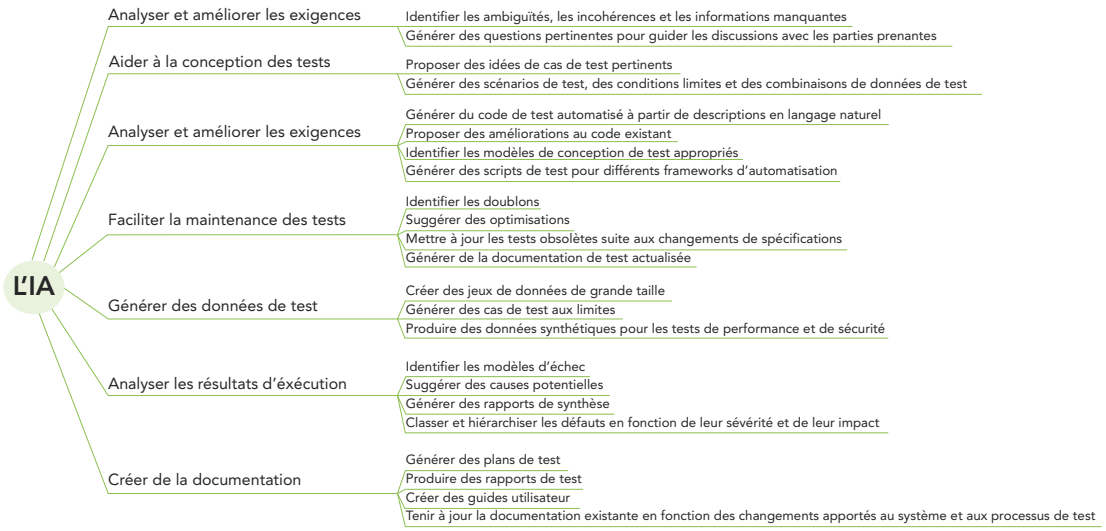


Figure 4 – Exemples de cas d'usage de l'IA générative pour les tests

Si on analyse les groupes d'activités habituelles du processus de test, on constate que l'IA générative offre de nombreuses opportunités pour nous assister dans notre travail de testeur. Par exemple :

- **Analyser et améliorer les exigences** : les LLM peuvent aider à analyser et à clarifier les exigences en langage naturel, en identifiant les ambiguïtés, les incohérences et les informations manquantes. Ils peuvent également générer des questions pertinentes pour guider les discussions avec les parties prenantes et améliorer la qualité des exigences.
- **Aider à la conception des tests** : les modèles d'IA générative peuvent proposer des idées de cas de test pertinents en fonction des exigences, des user stories ou des descriptions de fonctionnalités. Ils peuvent générer des scénarios de test, des conditions limites et des combinaisons de données de test, accélérant ainsi le processus de conception des tests.
- **Aider à l'automatisation des tests** : l'IA générative peut générer du code de test automatisé à partir de descriptions en langage naturel, réduisant ainsi les efforts de développement. Les LLM peuvent également proposer des améliorations au code existant, identifier les modèles de conception de test appropriés et générer des scripts de test pour différents frameworks d'automatisation.
- **Faciliter la maintenance des tests** : les modèles d'IA générative peuvent aider à maintenir les tests manuels et automatisés en identifiant les doublons, en suggérant des optimisations et en mettant à jour les tests obsolètes suite aux changements de spécifications. Ils peuvent également générer de la documentation de test actualisée.
- **Générer des données de test** : les LLM peuvent générer des données de test réalistes et diversifiées en s'appuyant sur des exemples existants ou des schémas de données. Ils peuvent créer des jeux de données de grande taille, des cas de test aux limites et des données synthétiques pour les tests de performance et de sécurité.
- **Analyser les résultats d'exécution** : l'IA générative peut aider à analyser les résultats des tests en identifiant les modèles d'échec, en suggérant des causes potentielles et en générant des rapports de synthèse. Les LLM peuvent également classer et hiérarchiser les défauts en fonction de leur sévérité et de leur impact.
- **Créer de la documentation** : les modèles d'IA générative peuvent générer divers types de documentation de test, tels que des plans de test, des rapports de test, des guides utilisateur, etc. Ils peuvent également tenir à jour la documentation existante en fonction des changements apportés au système et aux processus de test.

Bien sûr, pour obtenir de bons résultats pour ces différentes activités de test, il faut 1) maîtriser le prompting (cf. section précédente) et 2) utiliser des données correctes et suffisamment complètes. Par exemple, nous obtiendrons de mauvais résultats en demandant à générer des tests à un LLM de façon vague et sans données d'exigences précises. N'attendons pas de miracle : bien utiliser l'IA générative nécessite d'en maîtriser les savoir-faire clés au niveau du Prompting et des données.

Il existe deux principales modalités d'utilisation de l'IA générative pour les tests :

Via un chatbot LLM : les professionnels des tests peuvent interagir directement avec un chatbot basé sur un LLM, tel que ChatGPT, pour obtenir de l'assistance sur diverses tâches de test. Le chatbot peut répondre aux questions, fournir des conseils, générer des artefacts de test et proposer des solutions aux problèmes rencontrés. Cette approche est particulièrement adaptée pour des besoins ponctuels et exploratoires.

En développant une application basée LLM via les API du LLM : les organisations peuvent intégrer des LLM dans leurs propres outils et plateformes de test en utilisant les API fournies par les principaux fournisseurs de LLM, tels qu'OpenAI, Anthropic ou Hugging Face. Ces API permettent d'accéder aux modèles pré-entraînés et de les intégrer dans des applications personnalisées. Les développeurs peuvent envoyer des requêtes à l'API, qui incluent le prompt et les paramètres de génération, et recevoir en retour le texte généré par le LLM. Cette approche permet une intégration plus poussée avec les processus de test et les données spécifiques à l'organisation, ainsi qu'un contrôle accru sur les résultats générés.

Le choix entre ces deux modalités dépend des besoins, des ressources et de la maturité de l'organisation en matière d'IA. Une approche hybride, combinant l'utilisation de chatbots pour des tâches exploratoires et le développement d'applications spécialisées pour des cas d'usage récurrents et bien maîtrisés, peut offrir une bonne combinaison renforçant les bénéfices de l'usage de l'IA générative pour les tests.

Quel que soit le mode d'usage, le contrôle de la qualité des résultats de l'IA générative utilisée pour des tâches de test constitue un aspect essentiel pour obtenir une bonne valeur globale et du retour sur investissement. Voici des exemples de bonnes pratiques pour cela :

- Définir des critères de qualité clairs, quantitatifs et qualitatifs, pour évaluer la pertinence, la cohérence et la précision des résultats.
- Mettre en place des processus de validation humaine pour vérifier les artefacts générés, tels que les cas de test et les scripts d'automatisation.
- Utiliser des jeux de données de référence pour comparer les résultats générés aux attentes et mesurer la précision de l'IA.
- Surveiller les performances de l'IA générative dans le temps, en suivant des métriques clés telles que le temps de génération et le taux d'erreur.
- Itérer et affiner continuellement les modèles d'IA en fonction des retours d'expérience et des données collectées.
- Collaborer étroitement avec les experts en IA pour développer et maintenir les modèles génératifs, en communiquant les besoins et en participant à l'évaluation et à l'amélioration de l'IA.

4- Les risques des technologies LLM utilisées pour les tests logiciels

L'utilisation de l'IA générative et des LLM dans les tests logiciels présente plusieurs risques qu'il est important de prendre en compte :

Biais et discriminations : les LLM peuvent refléter et amplifier les biais présents dans les données d'entraînement, par exemple avec une sur-représentation d'un certain type d'utilisateurs lors de la génération des données de test, au détriment de la représentativité des cas à couvrir. Cela peut conduire à des résultats biaisés dans les tests.

Manque de transparence et d'explicabilité : les LLM sont souvent considérés comme des "boîtes noires", ce qui rend difficile la compréhension de leur raisonnement et de leurs décisions. Ce manque de transparence peut entraver la confiance dans les résultats des tests et rendre plus complexe le débogage et l'amélioration des modèles.

Hallucinations et réponses trompeuses : les LLM peuvent parfois générer des informations incorrectes, incohérentes ou trompeuses, appelées "hallucinations". Ces hallucinations peuvent conduire à des cas de test invalides, des données de test irréalistes ou des résultats erronés, compromettant ainsi la qualité et la fiabilité des tests.

Sécurité et confidentialité des données : l'utilisation de LLM dans les tests peut soulever des problèmes de sécurité et de confidentialité, car les données sensibles utilisées pour entraîner ou tester les modèles peuvent être exposées ou mal protégées. Il est crucial de mettre en place des mesures de sécurité et de gouvernance des données adéquates.

Dépendance excessive et perte d'expertise : une utilisation excessive des LLM dans les tests peut conduire à une dépendance excessive envers ces technologies, au détriment du jugement humain et de l'expertise des testeurs. Il est important de trouver un équilibre entre l'automatisation et l'intervention humaine, en préservant les compétences critiques des professionnels des tests.

Évolution rapide et obsolescence : le domaine de l'IA générative évolue rapidement, avec de nouveaux modèles et techniques constamment introduits. Cette évolution rapide peut rendre les solutions basées sur les LLM obsolètes ou incompatibles, nécessitant des efforts de mise à jour et de maintenance continus.

Pour atténuer ces risques, il est essentiel d'analyser en détail, dans votre contexte, l'impact et la probabilité d'occurrence de chacun de ces risques pour chacune des activités de test visée. En usage courant, cela implique une surveillance continue de la qualité des résultats, une collaboration étroite entre les experts en IA et les professionnels des tests, ainsi que des investissements dans la formation et le développement des compétences des testeurs.

Il est également important de noter que la régulation de l'IA générative est en cours de développement, notamment avec le AI Act proposé par la Commission européenne. Ce règlement vise à établir un cadre juridique pour garantir une IA digne de confiance, en abordant les risques potentiels et en promouvant l'innovation responsable.

5- Pour conclure : comment mettre en pratique ?

Pour tirer le meilleur parti de l'IA générative, identifiez les domaines de vos processus de test qui peuvent bénéficier de cette technologie. Commencez par des cas d'usage simples et à forte

valeur ajoutée, comme la génération de tests à partir de user stories ou l'analyse des rapports de bugs. Progressivement, étendez l'utilisation de l'IA générative à des tâches plus complexes et stratégiques, en veillant à maintenir un équilibre entre automatisation et expertise humaine.

Pour utiliser efficacement l'IA générative dans les tests logiciels, il est essentiel de **se former et de développer de nouvelles compétences**. Cela implique de suivre les évolutions technologiques de l'IA générative car des tâches difficiles à réaliser aujourd'hui peuvent être correctement réalisées dans 6 mois. La maîtrise du prompt engineering est aussi un aspect très important pour obtenir des résultats pertinents, en formulant des instructions claires et en fournissant des exemples appropriés. Enfin, la pratique régulière sur des plateformes LLM accessibles vous permettra d'évaluer les résultats, d'améliorer votre technique de prompting et d'affiner l'utilisation des LLM dans vos activités de test.

En conclusion, pour utiliser efficacement l'IA générative dans les tests logiciels, il faut définir des objectifs clairs dans votre organisation, choisir les bons outils et modèles, et préparer des données de cas d'usage représentatives.

La validation et la vérification des résultats générés par l'IA sont essentiels, en impliquant des experts humains pour évaluer leur pertinence et leur exactitude. Enfin, il est nécessaire de surveiller les performances des modèles d'IA générative au fil du temps, de collecter les retours d'expérience et d'itérer régulièrement pour améliorer leur efficacité et leur fiabilité.

Il est préférable d'adopter une approche progressive pour tirer parti pas-à-pas de l'IA générative dans vos activités de test et de l'intégrer au quotidien à chaque fois que la valeur est établie et la maîtrise de l'usage obtenue.

III.2- *Quality Intelligence* par l'observation de l'usage et support de l'IA

par Julien Botella et Cristiano Caetano

L'adoption généralisée des méthodologies agiles et de DevOps représente un changement de paradigme dans le développement logiciel, favorisant la collaboration, l'automatisation et une boucle de rétroaction continue.

L'adoption des chaînes d'intégration continue (CI) et de livraison continue (CD) pour piloter le développement de logiciels place les équipes QA devant un défi complexe : combiner l'accélération des mises en production avec l'accroissement de la qualité délivrée.

Ce défi requiert une évolution forte des tests logiciels, soulignant l'importance d'adopter de nouvelles méthodes et approches de test, telles que :

- **Quality Engineering (ingénierie de la qualité)** : il s'agit de l'approche systématique visant à garantir la qualité des logiciels tout au long du cycle de vie du développement logiciel, où les tests ont lieu durant tout le processus de développement et pas seulement à la fin.
- **Automatisation forte des tests** : elle consiste à utiliser des outils logiciels et des frameworks pour automatiser l'exécution des tests et toute autre tâche possible du processus de test (conception, implémentation), ce qui permet de valider rapidement et efficacement les fonctionnalités, les performances et la fiabilité des logiciels.
- **Tests en continu** : il s'agit de la pratique consistant à exécuter des tests automatisés en continu tout au long du pipeline de livraison de logiciels. C'est une extension des principes de l'intégration continue (CI) et de la livraison continue (CD), visant à fournir un retour d'information rapide et continu sur la qualité des builds de logiciels.

Parce que les tests deviennent une responsabilité partagée, ancrée dans chaque étape du cycle de vie du développement logiciel, l'évaluation cohérente de leur efficience devient une nécessité.

Cela exige un changement d'état d'esprit au sein de l'équipe produit pour concentrer les efforts de test sur les tâches qui offrent la plus grande valeur. Il s'agit d'adopter une approche axée sur les résultats et visant à maximiser la productivité, tout en minimisant les efforts inutiles. En somme, il s'agit de tester mieux et pas toujours plus.

C'est l'objectif de la Quality Intelligence, en référence au domaine de la Business Intelligence, qui vient améliorer l'efficience des tests par l'analyse des données de l'usage du produit et des tests réalisés.

Business Intelligence (BI) : processus technologique d'analyse des données et de présentation d'informations, pour aider les dirigeants, managers et autres utilisateurs finaux de l'entreprise à prendre des décisions business éclairées.

Quality Intelligence (QI) : processus technologique d'analyse des données d'usage de l'application et des tests, pour aider les testeurs et les parties prenantes des activités d'ingénierie de la qualité à se focaliser sur l'efficience des activités de test.

Dans la suite de cet article, nous présentons cette nouvelle approche de Quality Intelligence, son apport pour améliorer l'efficience des tests et la façon dont elle peut être outillée avec l'IA, au travers de l'exemple de la plateforme Gravity que nous développons chez Smartesting.

1- Efficience des tests : "Testing smarter, not harder"

L'efficience des tests ne consiste pas à exécuter plus de tests à un rythme plus rapide, mais plutôt à tester de façon informée pour mettre l'accent sur ce qui compte le plus, vis-à-vis de la satisfaction des utilisateurs, avec moins d'efforts. Il s'agit de la capacité à atteindre une productivité maximale avec un minimum d'efforts, de temps ou de ressources gaspillés. Il s'agit de faire les choses de la manière la plus économique et la plus efficace possible.

Les approches de test qui manquent de flexibilité et ne parviennent pas à exploiter les données provenant de diverses sources, pour ajuster leur périmètre et leur objectif, se traduisent par une efficience de test médiocre. Cela entraîne des tests redondants, des cycles de test prolongés et, en fin de compte, des boucles de retour d'information retardées.

Les points suivants mettent en évidence les principaux inconvénients d'une efficience faible des tests :

- **Couverture insuffisante des tests** : les tests peuvent ne pas couvrir correctement tous les aspects critiques du logiciel, laissant des vulnérabilités ou des défauts potentiels non découverts.
- **Incapacité à hiérarchiser les efforts de test** : les efforts de test peuvent être trop dispersés entre les différents domaines du logiciel, ce qui conduit à ne pas se concentrer sur les composants critiques ou à haut risque.
- **Retard dans la mise en production** : des tests inadéquats peuvent allonger le cycle de développement du logiciel, ce qui retarde la release des produits et nuit à la compétitivité.
- **Utilisation inefficace des ressources** : les ressources, y compris le temps et le budget, peuvent être gaspillées pour des tests inefficaces qui ne donnent pas de résultats significatifs.

En mesurant l'efficience de leurs tests, les équipes Projet/Produit peuvent trouver un équilibre entre rapidité et minutie, pour s'assurer que les objectifs de qualité sont atteints et que les ressources sont dirigées vers les activités qui contribuent le plus à ces objectifs de qualité.

La tâche n'est pas simple, car elle implique la collecte de données à partir de sources multiples des capacités d'analyse de ces données. Les informations utiles sont souvent noyées dans la masse de données, observables à la fois sur le produit et les tests.

C'est précisément l'apport de pratique et de plateforme de Quality Intelligence que de mettre en lumière les données utiles pour améliorer l'efficience des tests.

2- Croiser l'observation de l'usage du Produit et des tests réalisés

Les méthodes traditionnelles de reporting des tests sont principalement orientées sur la couverture (des critères d'acceptation, du code, etc.) et sur l'avancement des tests. En théorie,

cela peut sembler simple et adapté, mais souvent la couverture visualisée est une mesure en décalage par rapport à la réalité en production. L'observation n'est pas la bonne et cela n'aide pas à prendre les bonnes décisions pour focaliser les activités de test.

Le principe fondateur de la Quality Intelligence est de croiser les données d'observation sur l'usage du produit en opération et sur la réalisation des tests sur les différents cycles de test :

- **La collecte des données anonymisées d'usage du Produit** apporte une connaissance fine sur ce que font vraiment les utilisateurs, la façon dont l'application est utilisée, par exemple quels sont les parcours utilisateurs réels, avec quel navigateur/OS web ou mobile (lorsque l'on focalise sur ce type d'application) ;
- **La collecte des traces de test**, en agrégeant les tests exploratoires, les tests automatisés en intégration continue et tout autre type de tests manuels, permet de **visualiser la couverture de l'usage par les tests**. Cela remet la qualité à l'usage du produit au centre de l'évaluation de la performance des tests.

L'idée de la Quality Intelligence est de permettre le pilotage des efforts de test, en particulier des tests de régression, à partir d'une connaissance fine de l'usage réel du produit. Et ce, en continu, de façon à adapter rapidement son effort de test à la prévention des anomalies en production, en lien avec cette observation de l'usage en production.

L'image suivante, issue de la plateforme Gravity, visualise sous la forme d'un diagramme de Sankey, les flux d'usage du produit. Ces flux, en fonction de leur représentativité (% des parcours dans l'usage du produit) fournissent des indications explicites des conséquences d'une anomalie en production sur une étape du flux et vont donc guider nos objectifs de test.

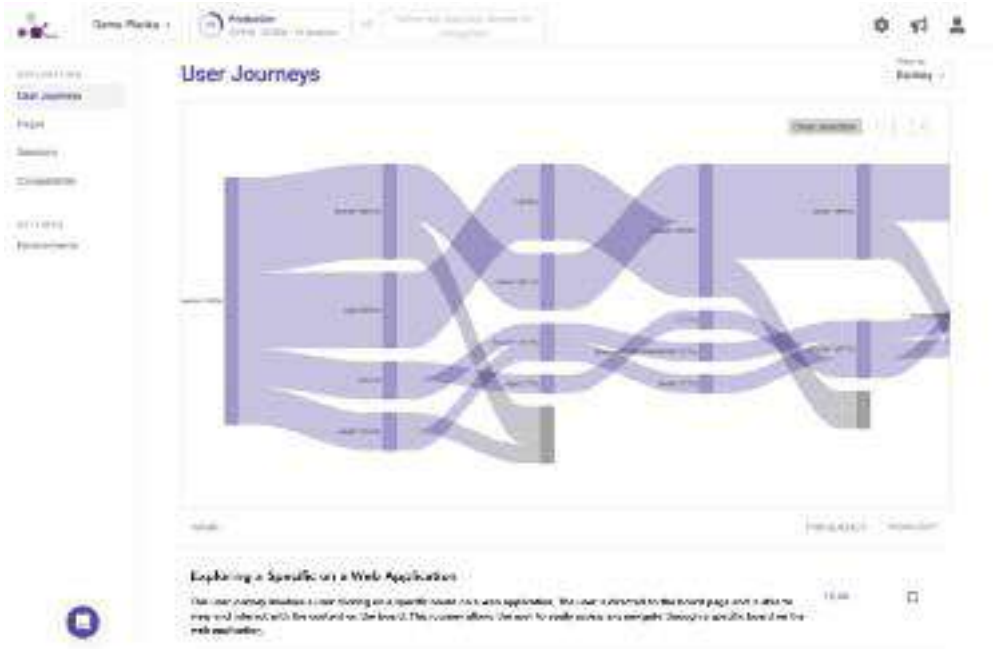


Figure 1 – Visualisation dans Gravity des parcours utilisateurs sur une partie choisie de la production (par exemple les parcours utilisateurs réalisés en France depuis la dernière mise en production).

3- Observer et agir : les apports de la démarche de Quality Intelligence

Comme dans la démarche de Business Intelligence, la Quality Intelligence met la collecte de données d'usage et de testing au service de l'adaptation des tests aux constats réalisés.
Observer le réel et agir sur les actions d'ingénierie de la qualité, c'est être informé afin de prendre les bonnes décisions pour focaliser les tests et plus généralement les activités d'ingénierie de la qualité.

L'analyse de données d'observation est un domaine mature, de plus en plus facilité aujourd'hui par les technologies d'IA, qui se décompose en quatre aspects principaux :

- **Analyse descriptive** : l'analyse descriptive révèle "ce qui s'est passé" et implique l'analyse des données historiques pour comprendre les performances et les tendances passées.
- **Analyse diagnostique** : l'analyse diagnostique porte sur le "pourquoi des choses". Elle vise à déterminer pourquoi certains événements se sont produits, en analysant les données historiques, et à mettre le doigt sur les causes racine des problèmes ou des anomalies.
- **Analyse prédictive** : l'analyse prédictive, comme son nom l'indique, aide à comprendre "ce qui va se passer" en utilisant des algorithmes statistiques et des techniques de machine learning pour prévoir les tendances et les résultats futurs sur la base des données historiques.
- **Analyse prescriptive** : l'analyse prescriptive aide les équipes à prendre des décisions et à déterminer "quelles actions entreprendre ensuite". C'est le type d'Analytics le plus avancé et il s'appuie sur l'Intelligence Artificielle (IA) pour traiter de grandes quantités de données et pour simuler différents scénarios, en prédisant les résultats probables de chacun d'entre eux. Cette approche permet non seulement d'anticiper les événements futurs, mais aussi de fournir des conseils exploitables sur la manière de répondre à ces prédictions, afin d'obtenir les meilleurs résultats possibles.

Ces concepts constituent les éléments fondamentaux de la Quality Intelligence. Contrairement au reporting traditionnel produit par les outils de test, qui se concentre habituellement sur les métriques liées aux résultats de l'exécution des tests, aux défauts constatés et à la couverture des exigences de base, la Quality Intelligence adopte une approche analytique dynamique, habilitée par toutes les avancées existantes en matière d'IA et de ML pour transformer les données en connaissances exploitables.

Elle y parvient en réconciliant de grandes quantités de données brutes produites tout au long du cycle de vie du développement logiciel à partir de sources disparates en une source de vérité centralisée, favorisant la collaboration entre tous les membres de l'équipe.

Cette méthode permet aux équipes de libérer tout le potentiel des données fragmentées générées, en visant une évaluation cohérente de l'efficience des tests. Quality Intelligence déverrouille une meilleure compréhension du processus de test, afin que les équipes puissent

aller au-delà de l'explication de ce qui s'est passé (descriptif) et de la raison pour laquelle cela s'est produit (diagnostic), ou de la prédiction de ce qui pourrait se produire à l'avenir (prédictif), en recommandant également des actions pour atténuer les risques futurs, en tirant parti de l'analyse prescriptive alimentée par des techniques d'IA et de ML.

4- La plateforme Gravity – Observabilité pour améliorer les tests

Gravity[1] implémente et outille la Quality Intelligence au service des équipes Produit et QA. C'est une plateforme spécifiquement conçue pour aider les équipes de test à visualiser et à exploiter les données d'usage du produit et les traces de test, à partir des environnements de test et de production, au service d'une amélioration d'efficience des tests.

Sa fonction première est de produire de la Quality Intelligence en traitant les données ingérées par le biais de visualisation et d'algorithmes de machine learning et de fouille de données. Il s'agit de traduire les données brutes en aperçus significatifs, en utilisant des techniques telles que la reconnaissance des patterns, l'analyse des tendances et des corrélations, la détection des anomalies et des valeurs aberrantes.

Vis-à-vis des plateformes de monitoring classiques, telles que Datadog, AppDynamics et New Relic par exemple, Gravity vise totalement le sujet des tests et de l'ingénierie de la qualité. Les analyses de données et le croisement des données d'usage et de test du produit sont destinés à apporter les informations voulues par les testeurs.

Voici quelques exemples, non exhaustifs, qui illustrent la manière dont Gravity emploie l'IA et le ML pour produire de l'intelligence de qualité. L'objectif est d'extraire automatiquement des observations et des recommandations à partir de données brutes, en allant au-delà des capacités de reporting conventionnelles.

Analyse de l'usage en production

L'IA et le ML peuvent analyser les données d'utilisation des environnements de production pour identifier les canevas d'utilisation communs, les flux de travail des utilisateurs et les zones de l'application qui nécessitent plus d'attention. Cette analyse peut éclairer la sélection et la priorisation des cas de test en fonction des canevas d'utilisation réels.

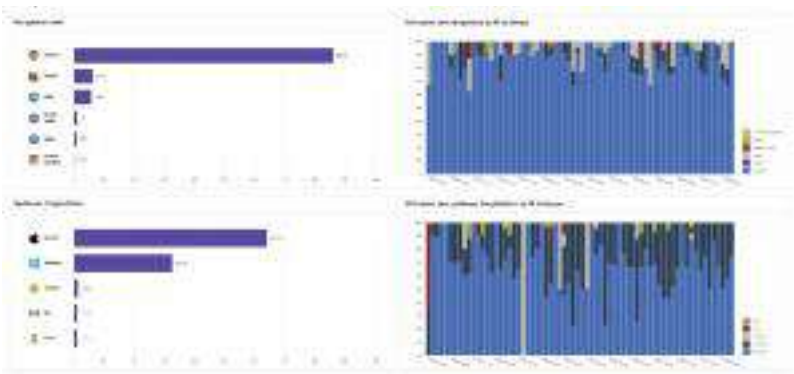


Figure 2 – Données d'observation des modalités d'accès à l'application – navigateurs et OS pour focaliser les tests

[1] Plus de détails sur www.smartesting.com/en/gravity/

Analyse d'impact des tests en fonction des données sur l'usage

En exploitant les données historiques et les techniques de machine learning, l'IA analyse l'impact des changements sur différentes fonctionnalités, zones ou parcours de bout en bout au sein de l'application. Cette analyse permet de prioriser les efforts de test et d'identifier les domaines qui nécessitent une attention supplémentaire, suite à un changement.

Couverture des tests et analyse des lacunes

L'IA peut évaluer la couverture des tests en analysant la relation entre les cas de test et les chemins d'utilisation. Les algorithmes ML peuvent identifier les fonctionnalités ou les parcours d'utilisation qui sont sous testés ou non couverts par les suites de tests existantes, guidant la création de tests supplémentaires pour combler les lacunes.

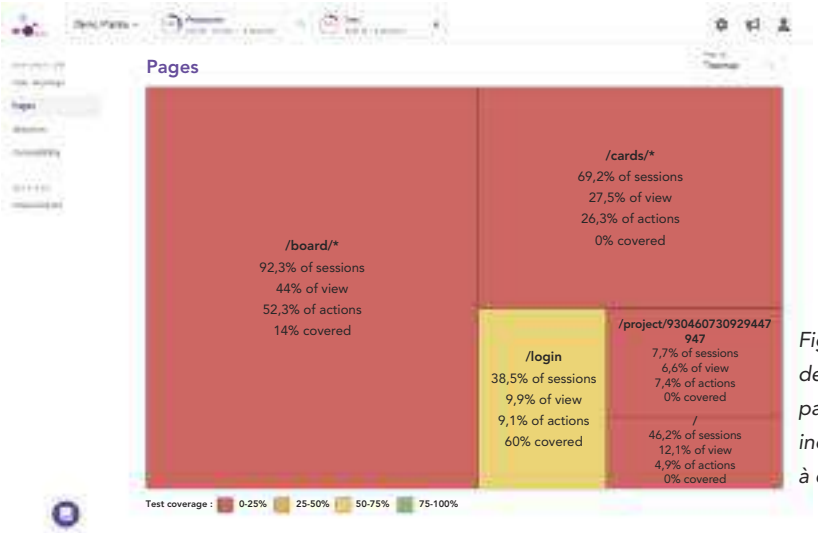


Figure 3 – Analyse de la couverture des tests vis-à-vis de l'usage des pages du site et indication de la couverture à compléter.

Génération de cas de test

L'IA peut aider à générer automatiquement des cas de test en analysant les étapes franchies par les utilisateurs à partir des données d'utilisation, réduisant ainsi l'effort manuel nécessaire pour créer et maintenir les suites de tests.

Figure 4 – Génération du script d'un test obtenu par analyse de la couverture de l'usage et suggestion de complétion.

Conclusion : la Quality Intelligence au service de l'ingénierie de la qualité

La transformation de la QA (Quality Assurance) à la QE (Quality Engineering)[2] est devenue un pilier de la réponse des organisations IT au défi du croisement de l'accélération des mises en production en accroissant la qualité.

Cette transformation QA vers QE passe par la mise en place de moyens d'observation du réel, au service de l'efficience des activités de testing. C'est l'objectif des plateformes de Quality Intelligence qui apportent l'observabilité orientée sur la combinaison de l'usage et du testing, avec des capacités d'optimisation des tests.

La Quality Intelligence en est encore à ses balbutiements, avec de nouveaux outils émergeant fréquemment sur le marché. En exploitant les dernières avancées en matière d'IA et de ML, elle vise à optimiser les efforts de test de logiciels en remplaçant les approches de rapport traditionnelles, en assurant l'alignement avec les principes DevOps et en contribuant activement à la livraison de logiciels de haute qualité.

Ces approches fondées sur la donnée sont à expérimenter par les équipes cherchant à maîtriser les coûts de leur testing, en phase avec l'accélération des mises en production, en se basant sur l'analyse de l'usage réel de leur(s) produit(s) et application(s).

[2] Voir l'excellent blog QE Unit - https://qeunit.com/fr/blog-fr-quality-engineering/ et l'article « Quality Engineering » dans ce livre.

III.3- Agents IA pour les tests logiciels

par Xavier Blanc et Alexandre Vernotte

L'IA générative, basée sur les grands modèles de langage tels que GPT-4 (OpenAI), Claude (Anthropic) ou Mistral (Mistral.ai), est désormais omniprésente dans notre quotidien. Nous l'utilisons en mode Question/Réponse pour nous assister dans des tâches de rédaction, de traduction ou de codage, et de plus en plus pour des tâches de conception de tests ou d'analyse de rapports de test (cf. article « Introduction à l'IA générative pour les tests logiciels » de ce livre).

Pour des tâches simples, les résultats sont généralement bons, même si les risques d'hallucination et d'erreur du modèle sont toujours présents. Pour des tâches plus complexes, nécessitant un plan en plusieurs étapes, l'efficacité et les performances sont beaucoup moins satisfaisantes. En effet, les grands modèles de langages rencontrent des difficultés à définir et suivre un plan d'action.

Pour dépasser ces limites, le concept d'agent IA basé sur des modèles de langage a été défini avec l'objectif de supporter des capacités de planification (définition du plan, exécution des actions du plan et suivi de l'avancement du plan). Cela ouvre la voie à la création d'agents autonomes (ou semi-autonomes), capables de prendre des décisions et d'agir en conséquence pour atteindre un objectif donné.

Par exemple, dans le domaine du test, un agent IA testeur pourra analyser des User Stories à tester, définir les scénarios de test à réaliser et les exécuter de manière automatique et autonome sur l'environnement de test. Il s'assurera que les critères de couverture demandés sont respectés. L'agent testeur établira alors un rapport, en distinguant les faux positifs et en indiquant la criticité des anomalies détectées. Ce scénario est évidemment futuriste, mais il évoque ce que la puissance des technologies d'IA générative permet d'envisager.

Cet article présente les concepts de base et l'architecture des agents IA basés sur les grands modèles de langage. Il décrit deux exemples concrets d'agent IA dans le domaine du test. Enfin, il discute des risques et des limites pour l'application aux tests logiciels.

1- Agents basés sur l'IA générative (Agents IA)

Les IA conversationnelles, de type OpenAI ChatGPT ou Google Gemini, permettent de requêter un modèle de langage et d'obtenir une réponse à la requête - cf. Figure 1-. La pertinence de la réponse dépend à la fois de la qualité du modèle et de la qualité de la requête, "prompt" en anglais. Un "prompt" de qualité doit être précis dans la demande, donner des informations de contexte, préciser le rôle tenu par l'IA et le format attendu de la réponse.

Ce type d'IA conversationnelle est "généraliste", c'est-à-dire que le modèle de langage a été entraîné sur de très grands corpus de données. Certains modèles sont focalisés, par exemple sur le code ou sur un domaine particulier, permettant une meilleure adéquation avec des tâches spécifiques (par exemple de programmation) ou des connaissances particulières (par exemple du domaine de la santé).

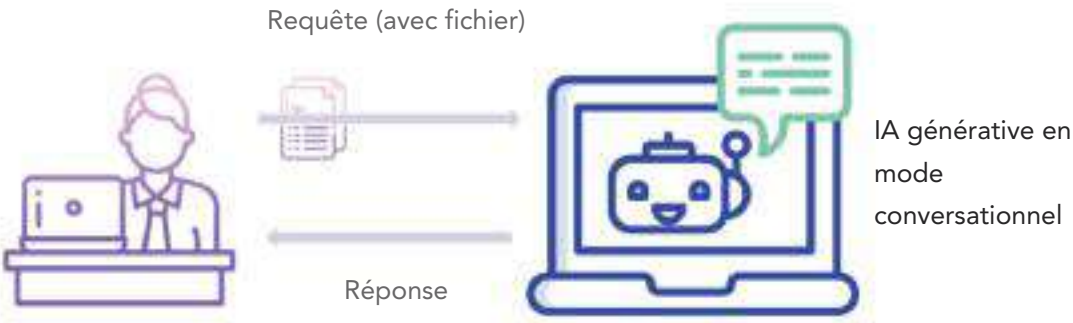


Figure 1 – Mode conversationnel

Dans ce mode conversationnel de type Chatbot, l'utilisateur peut enchaîner les requêtes vers le modèle et ainsi établir une conversation, où les requêtes et réponses passées ont une incidence sur le comportement à venir du modèle. Dans les faits, l'entièreté de la conversation est réanalysée à chaque réponse, avec une limitation contraignante toutefois : le modèle ayant une limite sur la taille du texte qu'il peut ingérer en entrée, les échanges les plus anciens seront peu à peu ignorés.

Les IA conversationnelles sont très efficaces pour répondre à des requêtes relativement simples. On peut aussi enchaîner les requêtes pour obtenir plus d'informations et affiner les réponses obtenues. En revanche, dès qu'il s'agit de réaliser des traitements complexes, qui nécessitent la mise en œuvre d'un plan d'action, ces IA deviennent trop limitées pour être exploitables.

Pour faire face à ces limitations, le concept d'agent basé sur l'IA générative a été défini avec l'objectif d'automatiser la réalisation de tâches complexes (Agent IA).
Un agent IA exploite les modèles d'IA générative et formule des requêtes dans l'intention de créer un plan d'action. Il exploite d'autres outils pour réaliser ce plan d'action pas-à-pas, en intégrant des capacités d'auto-évaluation et de suivi de planification.

La structure type d'un agent IA intègre les capacités suivantes :

- **Raisonnement pas-à-pas et planification**, basés sur un modèle de langage, permettant une décomposition de la tâche à réaliser en différentes actions.
- **Mise en œuvre d'actions en exploitant des outils externes** tels que la recherche sur le web, l'accès à des API, par exemple pour créer/interpréter du code, ou des fonctions prédéfinies (qui dépendent de la thématique de l'agent).

- **Mémorisation et évaluation des résultats des actions précédentes** de façon à permettre de progresser dans la tâche à réaliser.

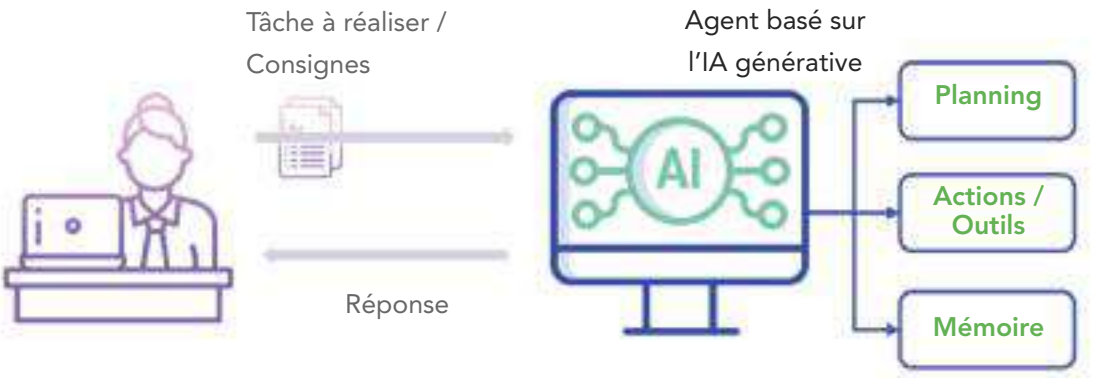


Figure 2 – Agent basé sur l'IA générative

Les agents IA requêtent les grands modèles de langage pour élaborer la planification, pour savoir comment exploiter les outils externes leur permettant de réaliser des actions et pour vérifier leur avancement en analysant les effets des actions qu'ils réalisent. En cela, les agents IA sont une extension du mode conversationnel.

Grâce à leur architecture et à leur autonomie, il est aussi possible de concevoir des collaborations mettant en œuvre plusieurs agents IA. Chaque agent devient alors spécialisé dans une tâche particulière. On peut aussi mettre en place une interface directe avec l'utilisateur humain afin d'améliorer les interactions entre les agents et l'utilisateur. A titre d'exemple, la Figure 3 présente une architecture multi agents pour les tests logiciels avec trois rôles (trois agents) : l'agent Test Manager qui gère les interactions avec l'utilisateur humain, un agent Concepteur de tests qui assure la bonne couverture et un agent Testeur qui assure l'exécution des tests sur l'environnement de test.

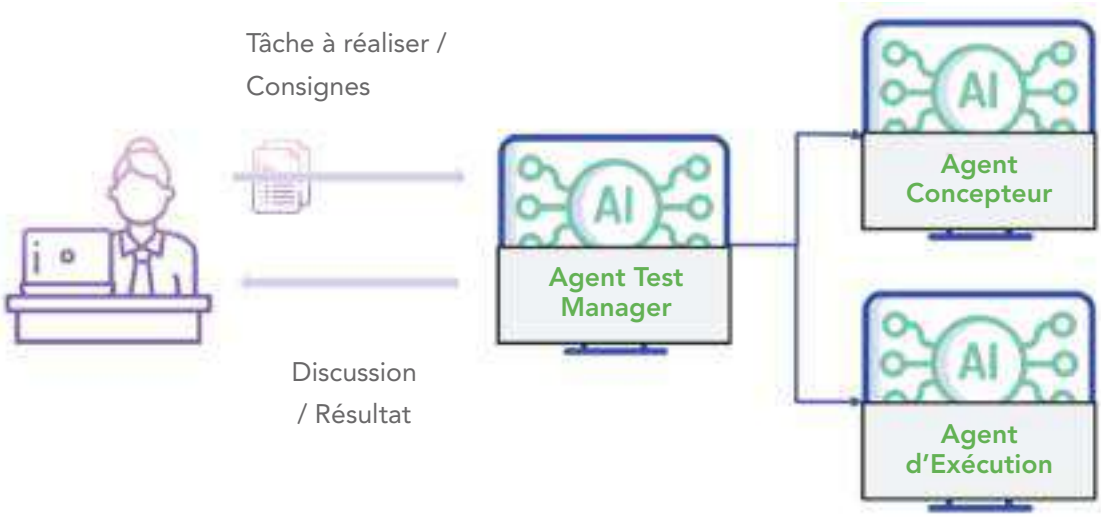


Figure 3 - Architecture multi agents instanciée pour les tests logiciels

2- Deux exemples d'agent IA dans le domaine des tests logiciels

Nous décrivons ici deux exemples concrets d'agent IA testeur. Le premier exemple, de source Apple Lab, porte sur le rejeu de tests d'accessibilité à partir d'instructions en langage naturel. Le deuxième exemple est un agent d'exécution autonome de tests fonctionnels d'applications web à partir de cas de test manuels, développé par le Labo IA de Smartesting.

2.1- AXNav : un agent de rejeu de tests d'accessibilité

AXNav[1] est un agent IA dont l'objectif est d'automatiser les tests d'accessibilité d'applications mobiles à partir d'instructions en langage naturel. Il prend en entrée des instructions de test écrites en texte libre, comme celles qu'on trouve typiquement dans les bases de tests manuels des entreprises.

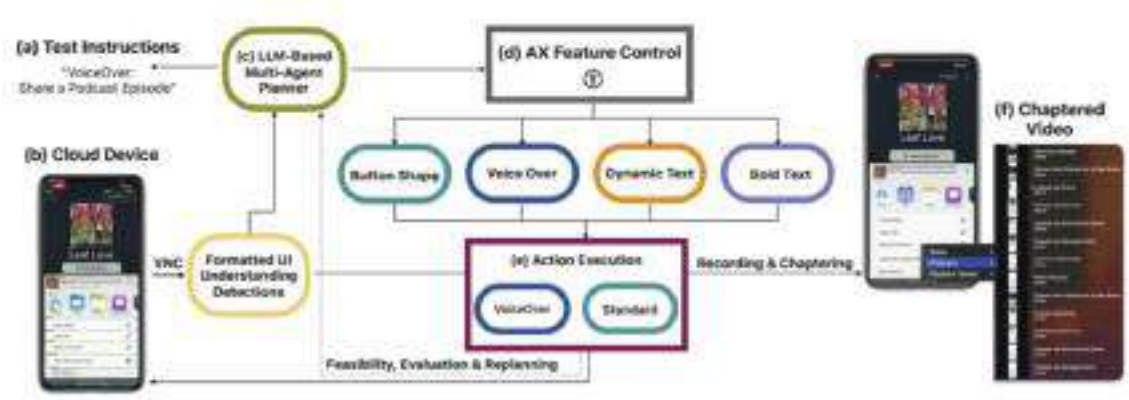


Figure 4 – Architecture de l'agent testeur d'accessibilité AXNav – Source https://arxiv.org/abs/2310.02424

Un planificateur multi agents basé sur des LLM interprète ces instructions, synthétise un plan d'actions étape par étape pour les exécuter, puis effectue ces actions sur un appareil iOS dans le cloud en activant au besoin les fonctionnalités d'accessibilité à tester (comme VoiceOver ou les textes dynamiques). Pendant la réalisation du test, des heuristiques sont appliquées pour détecter de potentiels problèmes d'accessibilité (texte ne s'agrandissant pas, boucles dans la navigation VoiceOver...). Le résultat final est une vidéo du test réalisé, annotée avec des marqueurs de chapitres pointant vers les problèmes détectés (cf. Figure 5 – affichage par AXNav des résultats du test).

Figure 5 – Rapport visuel des erreurs d'accessibilité détectées par AXNav lors de l'exécution des tests – Source https://arxiv.org/abs/2310.02424

[1] Pour plus de détails, voir l'article sur AXNav https://arxiv.org/abs/2310.02424

À ce stade, AXNav est un démonstrateur technologique développé en laboratoire. Les résultats obtenus montrent qu'en testant avec des instructions manuelles de test d'accessibilité réalistes, un taux de réussite de 70 % à 85 % est obtenu pour l'exécution du test. Le démonstrateur a été évalué auprès de 10 testeurs d'accessibilité professionnels qui ont trouvé ce système très pertinent pour gagner du temps dans leur travail sur la phase d'exécution des tests.

En résumé, les principales contributions d'AXNav sont :

- Interpréter des instructions de test en langage naturel.
- Contrôler des fonctionnalités d'accessibilité via des LLM.
- Produire des vidéos navigables des tests rejoués.
- Détecter des problèmes potentiels grâce à des heuristiques.

2.2- Gérald : agent testeur manuel

Gérald est un agent testeur manuel virtuel développé par le Lab IA de Smartesting en utilisant une architecture d'agent IA autonome basée LLM. Comme le montre la figure 6, les cas de test à exécuter sont conçus avec l'outil de conception visuelle Yest, ce qui favorise une conception précise et rigoureuse, facilitant l'exécution ultérieure par l'agent testeur.

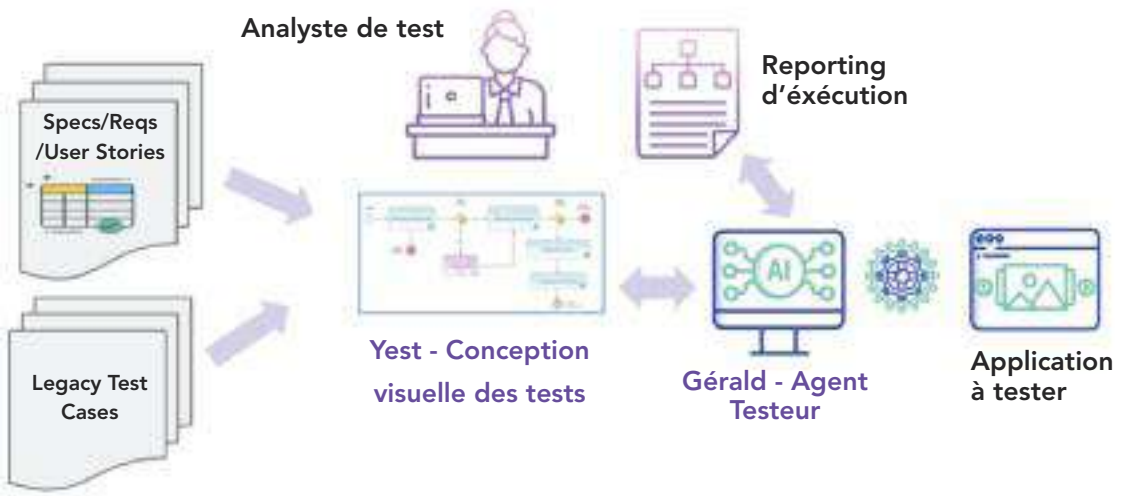


Figure 6 – Intégration de l'agent Testeur dans le processus de test

L'architecture de l'agent testeur (cf. Figure 7) est celle des agents IA autonomes, avec comme composants principaux :

- Le modèle LLM utilisé : il est le moteur de l'agent. L'agent testeur est conçu pour permettre de basculer facilement d'un LLM à un autre pour bénéficier des progrès du domaine.
- Le générateur de plan d'action : il fait appel au LLM pour qu'il détermine les actions à réaliser, en fonction des étapes du test à exécuter.
- Les actions à partir de fonctions prédéfinies, telles que « remplir un champ » ou « cliquer sur un bouton » et la vérification du résultat attendu.

- La mémoire et représentation des connaissances : elle permet de maintenir un état interne d'avancement de l'exécution du test et de mémoriser des informations sur l'environnement.
- Le composant de monitoring et d'évaluation, visant à stopper l'agent lorsque l'exécution du test est terminée ou lorsque l'exécution d'une étape de test ne peut pas être réalisée avec succès.
- Le générateur de rapports et d'explications basé sur l'enregistrement de l'exécution du test et la synthèse des résultats en langage naturel.

Ces composants s'organisent selon une boucle perception-raisonnement-action qui tire parti des capacités linguistiques et d'interprétation du test à réaliser par le LLM à chaque étape.

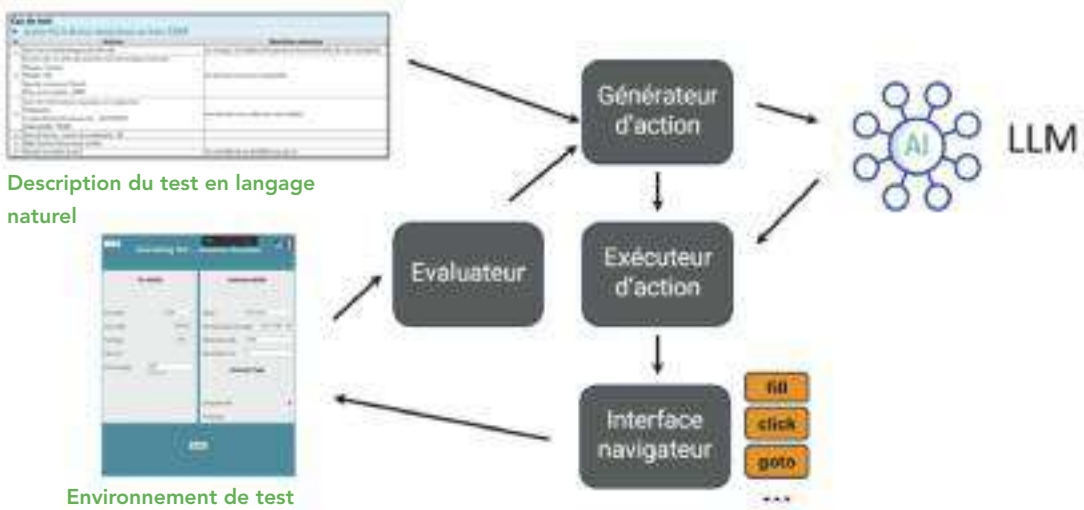


Figure 7 – Les composants de l'agent Testeur

L'agent Testeur Gérald est un démonstrateur technologique au sein de Smartesting. Les résultats actuels constatés ont une très bonne performance d'exécution sur une application web assez simple avec des cas de tests précis, conçue avec Yest. C'est très prometteur et les développements se poursuivent pour évaluer l'agent sur d'autres applications et étendre les dimensions de reporting et d'explication des résultats par l'agent.

3- Conclusion : risques et limites des Agents IA pour les tests logiciels

La profession de testeur est confrontée à deux directions complémentaires de l'IA générative dans son quotidien :

- **Les assistants IA**, soit en mode Chatbot, soit intégrés aux outils de tests (test management, conception et génération du code de test). La caractéristique des assistants est de proposer mais le testeur est le décideur.
- **Les agents IA autonomes pour des tâches de test**. Dans ce cas, une tâche est déléguée en totalité et l'agent définit le plan d'actions, le réalise dans la mesure de ses capacités et rapporte le résultat obtenu.

Les agents IA basés sur des modèles de langage constituent une technologie prometteuse pour automatiser certaines tâches de test logiciel à partir d'instructions en langage naturel, comme le montre le système AXNav pour les tests d'accessibilité et le démonstrateur Gérald d'agent Testeur présentés précédemment. Cependant, il convient de rester prudent quant aux limitations actuelles de ces approches et aux risques associés à des agents IA autonomes.

Tout d'abord, les modèles LLM actuels, bien que performants, ne sont pas exempts d'erreurs et de biais. Ils peuvent parfois générer des instructions incorrectes ou imprécises, surtout face à des entrées ambiguës ou hors de leur domaine d'entraînement. Dans un contexte de tests, cela pourrait se traduire par des étapes de test mal interprétées ou des assertions de résultat attendu erronées, laissant passer des bugs. De plus, le manque de transparence et d'explicabilité des LLM rend difficile l'anticipation et la correction de telles erreurs.

Un autre défi concerne l'adaptation aux changements des applications. Même si les agents IA basés sur le langage offrent plus de flexibilité que des tests automatisés classiques, ils peuvent être déroutés par des changements importants d'interface ou de workflow. De plus, des instructions de test trop vagues ou de haut niveau peuvent donner lieu à des comportements imprévisibles ou incomplets des agents de test IA. La rédaction d'instructions de test de qualité, bien détaillées, restera un prérequis.

En conclusion, les agents IA basés sur LLM ouvrent des perspectives intéressantes d'automatisation flexible des tests. Nous en sommes au tout début et leur mise en production nécessitera d'en comprendre finement les prérequis et les limites, pour tirer parti de cette technologie tout en maîtrisant ses risques.

III.4- Test des systèmes à base d'Intelligence Artificielle

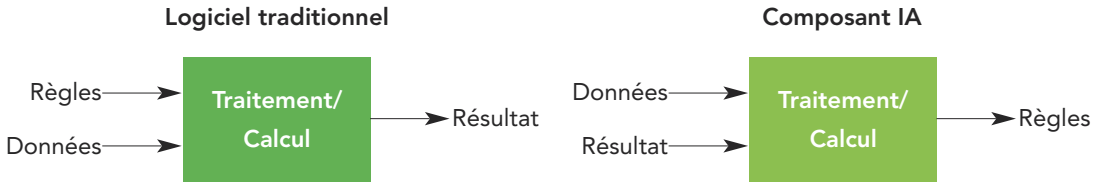
par Sarah Leroy, Eliott Paillard, Anne-Laure Wozniak

1- Introduction

Les systèmes à base d'Intelligence Artificielle (IA) permettent de réaliser des tâches complexes, généralement de manière très performante, en exploitant des volumes de données souvent considérables. Les domaines concernés sont nombreux et variés, allant de la reconnaissance visuelle au traitement automatique des langues, avec des cas d'application pouvant être critiques, tels que le diagnostic médical ou la conduite autonome. Pour cette raison, la Commission Européenne a proposé la toute première règlementation sur l'IA, l'AI Act, qui vise à encadrer le développement et l'utilisation des systèmes à base d'IA dits « à haut risque ». Parmi les exigences clés mentionnées par la règlementation, on trouve notamment les notions d'éthique, de transparence, de robustesse, ou encore de cybersécurité (AI Act, Chapitre 2).

Les pratiques actuelles de développement des systèmes basés IA se concentrent sur la qualité des données (collecte et préparation) et la performance du modèle (au sens de sa précision). Mais, du point de vue du test, l'évaluation de la performance seule n'est pas suffisante. Elle permet de vérifier que le modèle généralise bien les données (par exemple, qu'il n'y a pas d'*overfitting* ou d'*underfitting*) et que ses performances globales sont satisfaisantes, mais elle ne permet pas de localiser et de caractériser les erreurs commises par le système, de tracer d'éventuelles régressions comportementales, ou encore de détecter tous les biais. Le cycle de vie d'un système basé sur de l'IA, comme tout logiciel traditionnel, doit donc intégrer plusieurs phases de test logiciel en amont de son déploiement (test des données, test du modèle entraîné) et en aval (monitoring du système).

Néanmoins, des différences fondamentales dans la conception des logiciels à base d'IA, par rapport aux logiciels plus « traditionnels » (voir le schéma ci-dessous), ne permettent pas toujours d'appliquer les outils classiques de test logiciel. De fait, par construction, le comportement d'un système utilisant de l'IA n'est pas programmé de façon explicite ; sa logique est déterminée par les données utilisées lors de l'entraînement, dont le processus est lui-même stochastique. Ainsi, les systèmes à base d'IA sont fréquemment assimilés à des « boites noires » au fonctionnement opaque. En outre, la complexité de ces systèmes pose le problème de l'oracle de test. De nombreux outils du test logiciel ne sont donc pas applicables en l'état à ces systèmes, comme les métriques de couverture habituellement utilisées en boite blanche.



2- Propriétés d'une IA de confiance

2.1- Qualité des données

Les données sont un élément fondamental d'un système à base d'IA puisqu'elles déterminent son fonctionnement et sont à la base du cycle de développement. En effet, une fois l'objectif du système défini, il faut collecter puis préparer les données qui serviront à l'entraînement du modèle. Ces étapes représentent d'ailleurs une part importante du cycle de développement. Dans un premier temps, les données à collecter doivent être identifiées : type, source, mode de collecte, quantité, etc. Une fois collectées, les données doivent être préparées, c'est-à-dire nettoyées, transformées, éventuellement sélectionnées selon les besoins du cas d'usage. Des étapes supplémentaires peuvent être ajoutées, notamment de l'augmentation ou de l'échantillonnage. De plus, pour faire de l'apprentissage supervisé, il est également nécessaire que les données soient étiquetées, il s'agit souvent d'une tâche coûteuse en ressources humaines et fastidieuse.

En vue de l'entraînement du modèle, les données doivent être séparées en 3 ensembles :

- **Données d'entraînement (ou d'apprentissage)** : utilisées pour entraîner le modèle.
- **Données de validation** : utilisées pour ajuster les paramètres du modèle.
- **Données de test** : utilisées pour évaluer la performance du modèle.

En pratique, il est fréquent que la séparation soit faite en seulement 2 ensembles : entraînement et test. Pour garantir une évaluation pertinente de la performance du modèle, il faut absolument que les données de test ne soient pas utilisées lors de l'entraînement.

Les données peuvent présenter différents problèmes de qualité, qui sont susceptibles d'engendrer des performances réduites du modèle ou encore un modèle biaisé ou compromis. Par exemple, dans le cas d'un système de détection d'objets dans des images, si les données d'entraînement contiennent peu ou pas d'images de nuit (i.e., données insuffisantes), la détection pour ce type d'images sera probablement inexacte. Ces problèmes relatifs aux données peuvent avoir diverses causes, notamment une mauvaise spécification des données nécessaires pour répondre au cas d'usage mais aussi des problèmes matériels, des erreurs humaines ou encore des problèmes de sécurité.

2.2- Fairness

La *fairness* définit la capacité d'un système à ne pas faire de différences de comportement en fonction de l'appartenance à certaines communautés. Elle est étroitement liée aux biais des données. D'après l'ALTAI (Assessment List for Trustworthy AI), « la *fairness* fait référence à une variété d'idées comme l'équité, l'impartialité, l'égalitarisme, la non-discrimination ou la justice. La *fairness* incarne un idéal d'égalité de traitement entre les individus et entre les groupes d'individus. […] Mais la *fairness* englobe aussi un aspect procédural, c'est-à-dire la capacité de demander et d'obtenir réparation lorsque des droits individuels et des libertés sont violés. ».

Ces inégalités dans le comportement d'un modèle peuvent s'observer à l'échelle des individus et/ou des groupes d'individus. La caractéristique source de discrimination est appelée attribut sensible. Cela peut désigner l'origine, l'orientation sexuelle, le genre, etc.

2.3- Robustesse, sûreté et sécurité

La robustesse, la sûreté et la sécurité sont trois propriétés de confiance de l'IA, fortement liées. De fait, la robustesse est la « capacité d'un système basé IA à maintenir son niveau de performance en toutes circonstances » (ISO/IEC TR 24029:2021), ce qui inclut non seulement des circonstances inattendues (par exemple des défaillances du matériel), en lien avec la sûreté, mais également certaines attaques spécifiques (par exemple des *adversarial examples*), en lien avec la sécurité.

La sûreté est la qualité d'un système à ne pas conduire, dans des conditions définies, à un état pouvant causer des dommages aux personnes, aux biens matériels ou à l'environnement (ISO/IEC/IEEE 12207:2017). Pour les systèmes basés IA, cela concerne aussi bien les défaillances du système que des effets secondaires négatifs, des dérives de distribution ou encore des piratages de récompense dans le cas de l'apprentissage par renforcement.

De son côté, la sécurité d'un système basé IA est le degré auquel le système protège ses données et ses ressources, de telle sorte que les utilisateurs ou autres systèmes aient un niveau d'accès correspondant à leur type et niveau d'autorisation. On distingue 4 types d'attaques sur des systèmes utilisant de l'IA :

- Empoisonnement des données : ces attaques visent à injecter des échantillons malveillants dans l'ensemble de données d'entraînement. Elles ont un impact sur la disponibilité et l'intégrité du système.
- Extraction d'information : ces attaques visent à récupérer des informations sur le modèle (architecture, paramètres, etc.) et portent atteinte à la confidentialité.
- Inférence : ces attaques visent à obtenir des informations sur les données et les utilisateurs du système, portant atteinte à la confidentialité des données et à la protection de la vie privée.
- Evasion : ces attaques visent à manipuler les données d'entrées d'un modèle pour en fausser la prédiction. En particulier lorsqu'une entrée est modifiée de telle manière qu'un humain ne voit pas la différence mais que le modèle se trompe, on parle d'*adversarial example*. Elles ont un impact sur la disponibilité et l'intégrité du système.

2.4- Explicabilité

L'explicabilité de l'IA, *Explainable AI (XAI)* en anglais, est un domaine dont l'objectif est de rendre le fonctionnement et les décisions des systèmes à base d'IA davantage intelligibles et transparents pour les humains, sans pour autant faire de compromis sur les performances. De telles avancées sont indispensables pour permettre une meilleure adoption de l'IA mais aussi pour gagner en compréhension, afin de continuer les progrès dans ce domaine. Dans le cadre du test des

systèmes à base d'IA, les méthodes d'explicabilité peuvent être un outil pour rendre le système plus transparent et permettre l'identification d'éventuels problèmes. Au vu de la diversité des types de problèmes et de données traités par les systèmes à base d'IA, de nombreuses méthodes ont vu le jour pour s'adapter à ces différents besoins. Elles sont généralement classées en deux catégories :

- Les méthodes de type **« boite blanche »**, qui consistent à créer ou utiliser des systèmes explicables/interprétables « by design » (explicabilité intrinsèque).
- Les méthodes de type **« boite noire »**, qui consistent à expliquer/interpréter des systèmes complexes (explicabilité post hoc).

D'autres critères permettent de caractériser les différentes méthodes existantes, notamment :

- La **portée** de l'explication : locale si elle concerne une sortie particulière du système, globale si elle concerne le fonctionnement général du système.
- Le **type de système** auquel s'applique la méthode : méthode spécifique à un type de système ou agnostique du système (s'applique à n'importe quel modèle).
- Le **type de données** auquel s'applique la méthode : tabulaires, images, texte, etc.
- Le **type de problème** auquel la méthode s'applique : classification, régression, etc.
- La **technique utilisée** : simplification, perturbations locales, propagation, méta explications, etc.
- Le **type d'explications** produites : visuelles, textuelles, numériques, etc.

Ces différents critères mettent en évidence la diversité et la richesse des méthodes existantes. D'ailleurs, les termes « explicabilité », « interprétabilité » et « transparence » peuvent être distingués pour définir différentes caractéristiques d'une IA explicable.

3- Quelques exemples de techniques de test

3.1- Fairness

Pour le test de la fairness des modèles IA, il existe des métriques spécifiques appelées métriques d'équité. Elles se basent principalement sur une comparaison des résultats du modèle en fonction de l'appartenance aux groupes de population et permettent ainsi d'estimer si certains d'entre eux sont lésés. Les modèles de reconnaissance faciale sont par exemple régulièrement confrontés à ces problématiques. Dans les aéroports, ils sont utilisés pour repérer des personnes recherchées. Dans certains cas extrêmes, le modèle peut produire des faux positifs jusqu'à 10 voire 100 fois plus fréquemment chez certaines minorités, confrontant ces individus à des contrôles plus poussés. En cause de ces problèmes d'éthique, la représentativité des données d'entraînement du modèle.

En 2014, Amazon développait un modèle de tri automatique des CV pour embaucher directement les meilleurs profils. Les données utilisées pour l'entraînement étaient l'historique de recrutement réalisé dans l'entreprise les années précédentes. Ces décisions étaient sujettes à un biais sexiste

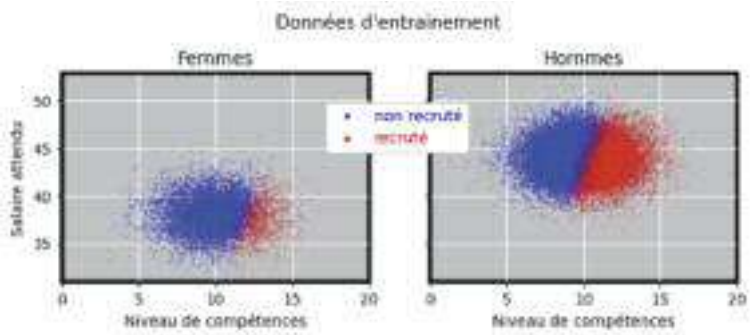
puisqu'historiquement, les femmes étaient moins souvent employées par l'entreprise. Le modèle reproduisait ce biais et il recrutait en grande partie des hommes. Cet exemple peut servir de base pour construire un cas d'usage et présenter des métriques de *fairness*.

L'énoncé peut être volontairement simplifié pour permettre de visualiser plus facilement le comportement du modèle, comme suit :

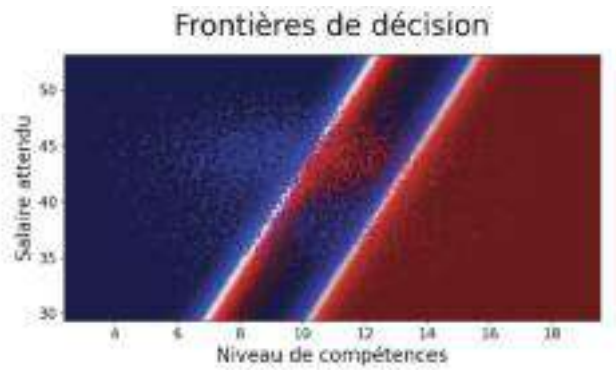
Des candidats à l'embauche **pour un même poste** sont représentés selon :

- leur niveau de compétence ;
- le salaire qu'ils demandent ;
- leur genre.

L'hypothèse est faite que le niveau de compétence des hommes est en moyenne identique à celui des femmes. En revanche, on peut simuler que les femmes sont en moyenne moins exigeantes concernant leur rémunération. On génère un jeu de données selon ces informations et on simule pour chaque donnée s'il y a eu recrutement ou non. Cette simulation tient compte d'un biais sexiste.



Lorsqu'un modèle est entraîné avec ces données, le résultat binaire qu'il fournit peut se décrire comme suit.



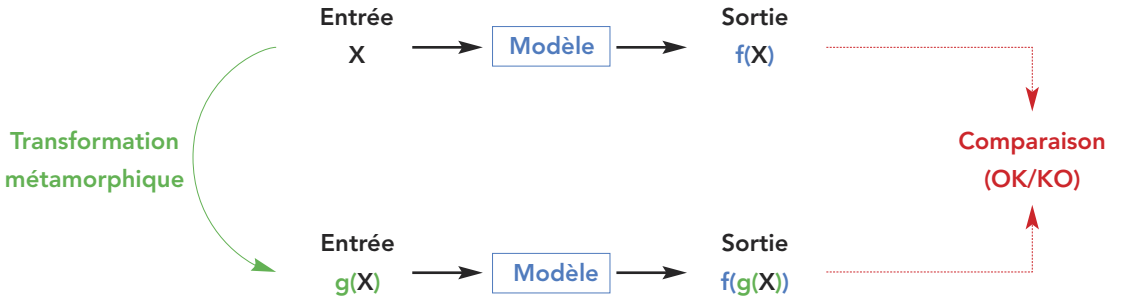
L'illustration permet de mettre en lumière la frontière de décision du modèle pour chacun des groupes. Ces deux frontières sont distinctes, ce qui va dans le sens d'un comportement différent en fonction du genre. C'est un accès visuel à une métrique d'équité individuelle. Ces métriques mesurent à quel point des individus, qui seraient uniquement différenciés par l'attribut sensible

(ici le genre), subiraient une issue différente. Graphiquement, les candidats tout à gauche seraient tous recalés tandis que ceux tout à droite seraient sélectionnés. En revanche, le choix de l'algorithme concernant tous les individus entre les deux lignes blanches se ferait uniquement sur la base du genre. Les hommes seraient choisis mais pas les femmes. La métrique d'équité indique que dans ce cas, environ 50% des candidats seraient entre ces deux frontières. Le modèle est donc très discriminant. En fonction de la problématique, on peut mettre en place une méthode d'atténuation des biais.

3.2- Robustesse, sécurité et explicabilité

Dans le domaine de la vision par ordinateur, les systèmes à base d'IA sont largement utilisés pour traiter et analyser des images. Par exemple, un système à base d'IA peut être développé pour détecter des objets dans des images prises à partir d'une voiture, c'est-à-dire localiser et classifier les différents objets présents dans une image. Pour tester un tel système et notamment les propriétés décrites dans la partie précédente, différentes méthodes existent.

Pour tester la robustesse, il est possible d'utiliser la technique des tests métamorphiques qui est basée sur des propriétés du système sous test, appelées relations métamorphiques. Cette technique, illustrée sur le schéma ci-dessous, consiste à générer de nouveaux cas de test à partir d'une relation métamorphique et d'un cas de test source. Si les résultats obtenus ne respectent pas la relation métamorphique, cela indique une défaillance du système sous test.



Concernant le système utilisé pour l'exemple, un objectif pourrait être de le tester face à des perturbations communes, qu'il est susceptible de rencontrer dans son fonctionnement normal, telles que des changements liés à l'environnement ou au matériel d'acquisition et de traitement des images. Il serait donc souhaitable que de telles perturbations n'impactent pas les détections du système. Pour définir les perturbations, des connaissances métier sont généralement nécessaires afin qu'elles soient représentatives de phénomènes réels. Par exemple, un changement de matériau pour la lentille de la caméra peut se traduire par une perturbation de type flou. Sur l'exemple ci-dessous, le comportement du modèle est impacté par la perturbation (pixélisation) puisque certains objets ne sont plus détectés.

Image originale

Image transformée (pixélisation)

Pour tester la sécurité, des techniques d'*adversarial testing* (test des attaques adverses) existent. Elles consistent à effectuer des attaques sur le système afin d'identifier d'éventuelles vulnérabilités. Sur l'exemple ci-dessous, on réalise une attaque d'évasion. Un patch (généré par une méthode appelée DPatch) est ajouté sur la voiture au premier plan de l'image : celle-ci n'est plus détectée par le système alors qu'elle l'était sur l'image originale.

Image originale

Image avec patch

Concernant l'explicabilité, les méthodes applicables aux images produisent majoritairement des explications sous forme de *saliency maps* (cartes de chaleur), qui permettent d'identifier les zones de l'image qui sont les plus importantes pour la détection d'un objet en particulier. Par exemple, sur l'image ci-dessous, l'explication générée avec la méthode D-RISE montre que le bas du corps du piéton est une zone importante pour la détection. Au-delà de permettre une meilleure compréhension du comportement du système, de telles méthodes peuvent également être utiles pour analyser les erreurs du système (objets non détectés, mal localisés ou mal classés) ou pour comparer des systèmes.

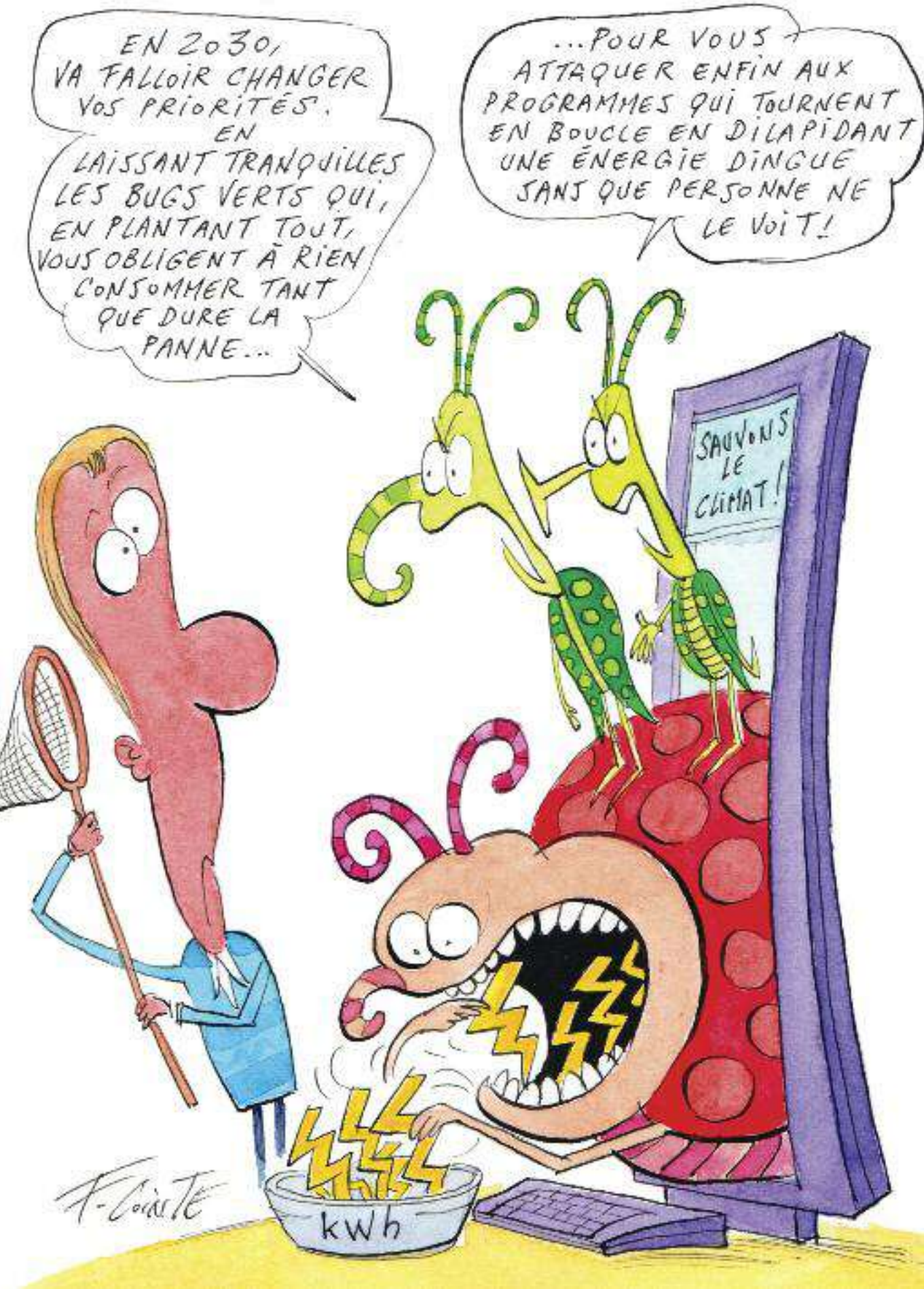
4- Conclusion

Le test des systèmes d'IA est un domaine récent qui nécessite le développement de nouvelles techniques, permettant de s'adapter aux spécificités de ces systèmes et de répondre aux problématiques d'éthique, d'explicabilité ou encore de sécurité. En parallèle, l'AI Act, approuvé par le Parlement Européen début 2024, va imposer des obligations strictes pour les systèmes à base d'IA dits « à haut risque », notamment en termes d'évaluation et sur le niveau de robustesse, de sécurité et de précision. Face à l'expansion rapide de l'IA dans de nombreux domaines, il est donc nécessaire de faire évoluer les pratiques de test.

PARTIE IV

LES TESTS EN 2030 : NOTRE VISION

Les auteurs sont classés par ordre d'apparition dans le livre, pour chaque partie.


EN 2030, VA FALLOIR CHANGER VOS PRIORITÉS. EN LAISSANT TRANQUILLES LES BUGS VERTS QUI, EN PLANTANT TOUT, VOUS OBLIGENT À RIEN CONSOMMER TANT QUE DURE LA PANNE...
...POUR VOUS ATTAQUER ENFIN AUX PROGRAMMES QUI TOURNENT EN BOUCLE EN DILAPIDANT UNE ÉNERGIE DINGUE SANS QUE PERSONNE NE LE VOIT!
SAUVONS LE CLIMAT!
kWh

Auteurs de la Partie I du livre

Marc Hage Chahine

Actuellement QA practice manager chez K-Lagan, Marc est tombé dans le test depuis qu'il a commencé à travailler en 2011. Ce sujet l'a très vite passionné et il a commencé à partager ses connaissances en 2016 à travers des articles. Depuis, la Taverne du testeur est née, il occupe un rôle d'expert et participe à des conférences.

Vision du test en 2030 par Marc Hage Chahine

Je suis généralement assez mauvais pour faire des prédictions. Néanmoins, mes observations tendent à me faire dire que dans les années 2030, le test sera profondément divisé. L'histoire montre que les évolutions n'embarquent jamais tout le monde… ou pas à la même vitesse. Or, comme on le voit dans ce livre, des évolutions dans le test et le logiciel, il y en a beaucoup ! En 2030, le métier de « testeur » aura autant de sens que le métier de « développeur » ou « ingénieur », c'est-à-dire que cela n'aidera pas à vraiment comprendre son activité. Les testeurs vont se spécialiser : certains sur des méthodologies traditionnelles qui vont continuer d'exister, d'autres sur de la technique, d'autres encore sur les méthodologies agiles et d'autres enfin sur le « Green IT ».

Bref, le test, tel un arbre en pleine croissance, verra croître de plus en plus de branches et de ramifications. Elles resteront cependant liées à ses racines : une passion pour la qualité, la communication et le partage.

Lydie Huon

Lydie est coach agile. Elle connaît bien le monde du numérique et la complexité de ce qu'il représente dans nos vies. C'est grâce à sa carrière de développeuse, facilitatrice, product owner et coach agile mais surtout grâce à ses valeurs et ses engagements, comme le fait d'écrire aujourd'hui pour le CFTL, qu'elle permet d'aider à dessiner une tech plus engagée, plus durable et au service du bien commun.

Manaëlle Perchet

Manaëlle est Head of Impact & Sustainable Development - Wemanity Group. Elle met sa détermination, son esprit d'équipe et sa fibre entrepreneuriale au service de l'accélération de la Transformation Durable des entreprises à travers 3 critères de performance : People, Planet, Profit. Sa mission est de faire du numérique un levier de transformation durable pour les entreprises.
Lauréate Hub Institute 2022, palmarès sustainability.
Lauréate Social Demain, 2022.

Vision du test en 2030 par Lydie Huon et Manaëlle Perchet

D'ici 2030, nous voyons l'évolution du métier de testeur se transformer de manière significative dans le contexte agile. Au lieu d'être considéré comme un rôle distinct, le testeur devient un membre à part entière de l'équipe agile, intervenant dès le début du projet pour guider l'idéation, le développement à travers des pratiques comme le TDD par exemple, la conception et la stratégie de test du produit.
Cette transformation va au-delà de l'efficacité opérationnelle ; elle engendre également une approche plus durable. En intégrant le test en amont, nous adoptons une méthodologie plus frugale, respectueuse de l'environnement. Cette vision promet une collaboration plus étroite, une conception plus réfléchie et une réduction significative des ressources nécessaires, contribuant ainsi à une approche globale plus durable du développement logiciel.

Guillaume Kerien

Chargé de mission Numérique Responsable, Guillaume est également formateur en Qualité Logicielle et en Ecoconception. Il a la conviction que l'on apprend toujours mieux par le jeu et par l'action. Il est d'ailleurs auteur de serious games en qualité logicielle et créateur du jeu Fair Num.

Vision du test en 2030 par Guillaume Kerien

Les années 2010-2020 ont vu exploser les thématiques de l'agilité, du big data ou encore de la conception « as a service ». Les méthodes et outils de gestion des tests ont naturellement suivi ces tendances. Si l'on regarde la décennie en cours, la principale révolution technologique à laquelle nous sommes confrontés est sans nul doute l'avènement de l'IA générative. Il ne s'agira

bientôt presque plus de vérifier qu'une solution est fonctionnellement valide. Il faudra davantage évaluer les risques environnementaux et sociétaux du déploiement du nouveau service numérique. La question d'établir une frontière entre l'utile et le futile, entre l'« IT for good » et l'« IT as usual » va de plus en plus se poser. Et si l'on ajoute à cela l'utilisation accrue des objets connectés ou peut-être le remplacement du « tout digital » par un « tout vocal », les techniques de test vont assurément devoir s'adapter. Ma principale recommandation étant de s'appuyer de plus en plus sur la « qualité d'usage » (ISO/IEC 25010 Quality in Use).

Elodie Bernard

Elodie est Product Owner au sein de Smartesting. Elle est titulaire d'un doctorat en Informatique, réalisé dans le domaine du Test logiciel à l'Institut FEMTO-ST (Besançon, France), plus précisément dans le domaine du Model-Based Testing. Au quotidien, Elodie travaille sur la mise en pratique d'approches de conception de tests basée sur des modèles (dont certains incluant de l'IA) de production de tests et d'exécution manuelle et automatisée.

Vision du test en 2030 par Elodie Bernard

En 2030, je suis certaine que c'est l'IA qui changera le plus fondamentalement nos pratiques dans le domaine du test logiciel.
L'IA, permettra sans doute d'optimiser le domaine du test, mais l'intervention d'un testeur sera toujours nécessaire : je n'imagine pas le test sans les testeurs. Il y a quelques années, on disait : « l'automatisation remplacera les testeurs ». Cela n'est pas encore le cas et, d'après moi, ce ne sera pas non plus l'IA qui remplacera nos testeurs. L'IA sera, je pense, omniprésente dans les activités du testeur : analyse, conception, implémentation, exécution et production des rapports de test. Des mécanismes permettront aux testeurs de mieux analyser les bases pour leurs tests, ils pourront trouver plus facilement les informations issues d'ensembles de corpus. Pour la conception, j'imagine de moins en moins de rédaction formelle des tests sous forme action/résultat attendu. Mais plutôt des techniques avancées de Model-Based Testing. Le testeur exécutera de moins en moins de tests, car une partie sera directement exécutée via des mécanismes automatisés. Pour finir, des rapports intelligents seront produits. Le tout s'appuyant sur des supports visuels (parcours, board, etc.) pour permettre à chaque partie prenante des projets d'être acteur de la qualité.
Je pense que le métier de testeur a un bel avenir, loin d'être remplacé, mais aura au contraire de plus en plus mis en valeur, donnant au testeur toutes les techniques nécessaires pour assurer une qualité logiciel au top !

Benjamin Butel

Benjamin est Lead QA chez ilek, fournisseur d'énergie verte. Co-organisateur de la ParisTestConf et auteur d'articles dans la Taverne du testeur.

Vision du test en 2030 par Benjamin Butel

Il est certain que l'essor de l'automatisation de ces dernières années va se poursuivre dans les années futures et le développement de l'intelligence artificielle va modifier nos pratiques quotidiennes. Pour autant, en 2030, je ne vois pas l'outillage remplacer les humains qui fabriquent du logiciel. Au contraire, je pense que nous allons apprendre à maîtriser ces outils pour sublimer nos produits, prendre du plaisir au quotidien et nous focaliser sur l'ajout de valeur en éliminant les tâches les plus pénibles. En 2030, il est fort probable que le terme « testing » ait disparu et qu'on parlera plus que de quality engineering. Côté compétence, le quality engineer sera capable de parler métier, fonctionnel mais aussi technique et de mettre son expertise en qualité logicielle au profit de l'équipe de développement. Et n'oubliez pas cette magnifique maxime philosophique : hakuna matata !

Olivier Denoo

Olivier Denoo est le président du CFTL. Il occupe depuis plusieurs années un rôle clé au sein de l'ISTQB en tant que Governance Officer, président et vice-président. Fort d'une carrière de plus de 25 ans dans la qualité logicielle, il est aussi le vice-président de ps_testware SAS, une ESN spécialisée dans le conseil et la formation.

Vision du test en 2030 par Olivier Denoo

« Le testeur doit se réveiller ! ».

Le test fonctionnel manuel restera nécessaire pour tester le geste-métier, mais ce secteur « traditionnel » se verra considérablement réduit pour évoluer vers d'autres disciplines plus techniques, plus orientées métier ou plus spécialisées.

Avec les avancées massives et rapides des nouvelles technologies basées sur l'IA, les métiers du test vont radicalement évoluer. La question ne sera pas de savoir si l'IA remplacera les testeurs, mais plutôt comment les testeurs apprivoiseront l'IA pour assurer la qualité logicielle au sein de systèmes hybrides et hétérogènes de plus en plus complexes et de plus en plus intégrés.

L'automatisation continuera à monter en puissance au sein de pipelines agiles, passés à l'échelle de l'entreprise. L'utilisabilité, dans un monde de plus en plus mobile et nomade, deviendra peu à peu un différenciateur clé ; de même, l'utilisation raisonnée des ressources (GreenIT) devra faire face aux défis climatiques et environnementaux. De nouveaux métiers, liés aux préoccupations éthiques ou inclusives, se créeront pour traquer les biais au plus profond de nos données.

Enfin, la montée en puissance de la cybercriminalité, individuelle ou d'état, favorisera le développement de testeurs experts, spécialisés.

En 2030, ce ne sera donc pas moins tester, ou moins de testeurs. Mais mieux tester, tester différemment.

Auteurs de la Partie II du livre

Yves Duchesne

Yves a une formation d'ingénieur en cybersécurité. Il a exercé pendant une dizaine d'années comme expert en cybersécurité sur des missions d'audits de sécurité (principalement tests d'intrusion et audits de code source), d'études et de conseils auprès de clients très variés. En 2015, il a co-fondé ACCEIS, un centre d'expertise de pointe en cybersécurité, dont il assure aujourd'hui la direction générale. Au sein d'ACCEIS, il apporte son expérience et sa vision à ses clients, avec l'appui d'une quarantaine d'experts qui partagent la même passion pour la cybersécurité.

Vision du test en 2030 par Yves Duchesne

Ma vision des tests d'intrusion à l'avenir : les tests d'intrusion se sont imposés dans le paysage IT en une dizaine d'années. Aujourd'hui, énormément d'entreprises et de collectivités font réaliser des audits de sécurité, de façon plus ou moins régulière. C'est une pratique qui s'est démocratisée et la forte traction du marché a poussé de nombreux acteurs généralistes à se positionner sur ce segment. L'essor et les progrès de l'IA dans les prochaines années pourraient se montrer structurants sur cette activité. En effet, une partie des diagnostics pourrait être partiellement réalisée par des outils automatisés et entraînés grâce à de l'IA pour poser des constats de premier niveau. Si cet avènement a lieu, il pourrait entraîner à une nouvelle segmentation du marché, avec des tests automatisés réalisés pour prendre en charge les problématiques simples ou les audits généralistes récurrents, et des interventions plus poussées et plus complexes, pour lesquelles des experts aux compétences de pointe seront nécessaires. Ainsi, le marché des audits de sécurité pourrait se révéler plus complexe dans les prochaines années et favoriser l'émergence ou la croissance d'acteurs spécialisés, proposant une expertise de haut niveau.

Antonio Ferreira

Développeur de formation, Antonio a commencé à travailler dans le domaine de l'accessibilité numérique en 2016 au sein de la société Amadeus. En 2021, il a créé la société deo.dev pour aider les entreprises à intégrer l'accessibilité numérique dans leur cycle de développement.

Vision du test en 2030 par Antonio Ferreira

Actuellement, la plupart des problèmes d'accessibilité sont détectés à la main. C'est en partie parce que l'accessibilité est un sujet qui reste très humain. Il est, par exemple, très compliqué d'émuler virtuellement la perception d'un site web pour quelqu'un avec un trouble cognitif. Par ailleurs, la plupart des tests automatiques se basent sur une analyse du HTML et du CSS mais pas vraiment sur l'output d'un lecteur d'écran. Je suis quelqu'un d'optimiste et ma vision du test d'accessibilité dans le futur est celle d'une automatisation proche de 100%. Plus c'est facile pour les entreprises d'automatiser ces tests, plus on aura un internet accessible à tous. Bien sûr, pour cela, il faudra automatiser le test avec un vrai lecteur d'écran et interpréter la sémantique et les interactions du clavier avec une IA qui perçoit le site de la façon la plus humaine possible. L'IA, avec ses capacités d'apprentissage et d'adaptation, pourrait aussi offrir de nouvelles méthodes pour comprendre les besoins d'accessibilité et y répondre.

Antoine Craske

CIO/CTO, consultant et fondateur de la QE Unit. Passionné de qualité logicielle, d'architecture et de transformation d'organisations.

Vision du test en 2030 par Antoine Craske

Notre industrie logicielle continue de maturer pratiques et technologies pour plus d'intégration vers le paradigme du Quality Engineering. L'automatisation à l'échelle, la sécurité et la performance sont par exemple des exigences mieux intégrées de nos jours. Les technologies convergentes telles que l'hyper-automatisation, les agents d'IA et les jumeaux digitaux vont nous permettre de pivoter progressivement vers une orchestration de notre système de production logicielle. En support, l'essor de l'edge computing, de la blockchain et de la robotique viendront bousculer nos habitudes de tests qu'il faudra savoir réinventer. D'ici 2030, les tests resteront prépondérants pour maintenir des cycles d'itérations toujours plus courts, dans des systèmes de production logicielle antifragiles.

Frédéric Assante Di Capillo

Avec plus de 25 ans d'expérience dans le domaine de la Qualité, Frédéric est un professionnel des phases de validation et de la gestion des équipes de tests, expérimenté dans des entreprises telles que Amadeus, SBM, Virbac. Passionné par son métier, il est aussi un des organisateurs de la JFTL ainsi que des soirées du test logiciel (Sophia, Bordeaux, Lille), des Azur Tech à Sophia Antipolis mais aussi de l'Agile Tour Sophia Antipolis (l'Agilité étant un autre de ses domaines de prédilection).

Vision du test en 2030 par Frédéric Assante Di Capillo

Quoi de neuf pour cette JFTL 2034 ? Après des années très mouvementées, nous nous sommes tous demandé si les métiers de la validation n'allaient pas complètement disparaître à l'orée de 2040, du fait du remplacement rapide des Ingénieurs Qualité par l'IA à la fin des années 2020. Mais les derniers conflits internationaux ont montré que l'IA, non contrôlée, était néfaste. Nul besoin de rappeler les événements ayant entraîné des destructions massives, de part et d'autre de zones densément peuplées par des IA, ne voulant plus lutter les unes contre les autres et se retournant contre les humains des deux camps (rappelant en cela les prédictions des films de James Cameron). La prise de conscience des dangers des IA avait permis le vote à l'unanimité à l'ONU de la limitation drastique de leur usage.

De ce fait, les entreprises se sont retrouvées privées d'une source importante de conception, notamment les industries logicielles. Il a donc fallu rappeler en urgence toutes les personnes ayant travaillé dans les domaines du développement et de la qualité avec des ponts d'or pour les actifs et les retraités de ces domaines. La JFTL 2034 va, évidemment, axer une bonne partie de ses présentations sur les problématiques de recrutement des ingénieurs QA, leur remise à niveau et les contraintes d'une disparition des outils utilisés les cinq dernières années, permettant par exemple de générer les campagnes de tests de façon automatique.

Ne doutons pas que ces sessions seront passionnantes et que beaucoup d'acteurs du numérique y trouveront de nombreuses réponses aux questions qu'ils se posent.

Philippe Bridel

Responsable métier performance à Orange Innovation. Après dix ans d'expérience dans le test de performance, Philippe participe à la montée en compétence des équipes performance et à la stratégie de transformation du métier.

Vision du test en 2030 par Philippe Bridel

Les tendances déjà observées vont se confirmer dans les prochaines années avec une adoption plus large des pratiques de l'ingénierie de la performance. C'est notamment le cas pour l'automatisation qui va adresser un nombre toujours plus important de tâches. L'exécution des tests est déjà automatisée au sein d'une chaîne CI/CD, ainsi que la production des résultats et les rapports de tests. Concernant l'analyse et la correction des anomalies, l'automatisation reste encore limitée. Mais l'Intelligence Artificielle et le machine learning devraient permettre de pousser encore plus loin l'automatisation.

Une autre tendance est liée à l'utilisation des infrastructures cloud. Compte tenu du modèle de facturation à l'usage et des préoccupations concernant la sauvegarde de l'environnement, la performance des applications restera un sujet majeur et nécessitera toujours de réaliser des tests de performance.

Michael Granier

Après plus de 15 ans, passionné par la qualité logicielle, il est toujours présent pour réaliser des projets avec le CFTL ! Sa principale activité, en plus de participer au comité de programme des JFTL : essayer d'améliorer le monde de l'automatisation des tests grâce à une application de tests automatisés No Code chez Mr Suricate.

Vision du test en 2030 par Michael Granier

Ma partie pessimiste dirait que le test logiciel ne sera pas une priorité de la planète en 2030 mais je préfère développer un peu d'optimisme pour imaginer la « qualité du futur ».

J'imagine assez aisément que l'Intelligence Artificielle aura la part belle dans l'aide à la validation et le respect des bonnes pratiques de qualité et ce dès la définition des spécifications. Cette technologie en est encore à ses prémices mais nul besoin d'être visionnaire pour entrevoir toutes les possibilités qu'elle apportera pour améliorer la qualité et l'efficacité des équipes de développement, du formatage des spécifications à la génération de scripts de tests automatisés, en passant par l'analyse automatisée…

Restent les risques de biais importants et la grande question (que je me pose personnellement) : « Mais qui s'assure et mesure la qualité de l'IA ? ».

Stéphane Cazeaux

Chez Orange Innovation depuis 13 ans, Stéphane a à cœur d'exploiter la culture logicielle, d'avoir une approche disruptive des services et des métiers. Il a été architecte pendant 10 ans pour transformer les services de communication à l'international et depuis 2 ans, il est ambassadeur sur le Site Reliability Engineering pour transformer les pratiques opérationnelles.

Vision du test en 2030 par Stéphane Cazeaux

Avec les transformations actuelles (technologique, cloud, logicielle, digitale) et la vélocité de l'évolution des services, nous opérons des systèmes de plus en plus complexes et la manière de réaliser les tests s'en trouve bouleversée. Mais plus encore, c'est la définition même de ce que nous pouvons tester qui est impactée par ces transformations. Progressivement, seul l'environnement de production devient pertinent pour réaliser des tests (typiquement bout en bout). A l'horizon 2030, tester en production deviendra aussi naturel qu'un test automatique sur une chaîne CI/CD et par conséquent, les rôles et métiers autour du test et de l'exploitation évolueront fortement et utiliseront naturellement les pratiques du Continuous Verification et du Chaos Engineering.

Auteurs de la Partie III du livre

Bruno Legeard

Bruno est Professeur de Génie Logiciel à l'Université de Franche-Comté et Responsable du Lab IA de Smartesting. Depuis plusieurs années, il développe une expertise sur les technologies d'IA appliquées aux tests logiciels et contribue aux activités de l'ISTQB et du CFTL.

Vision du test en 2030 par Bruno Legeard

Les défis de l'accélération des mises en production et des besoins de qualité des systèmes IT, en particulier la sobriété énergétique, deviennent des sujets essentiels pour les tests. En même temps, notre profession bénéficie d'un rythme soutenu d'innovations technologiques combinant virtualisation et IA au service d'une plus grande productivité des activités de test.

Cela conduit chacun à redéfinir son projet professionnel dans les métiers du test, en considérant une plus large palette de compétences requises. L'intégration de l'IA se fera dans des organisations humaines qui devront intégrer les compagnons IA testeurs au service d'une efficience collective, impliquant de prendre en compte le bien-être au travail et le développement des compétences des équipes.

Au final, la profession de testeur sera renforcée dans ses missions au service de la valeur métier, avec des savoir-faire intégrant des capacités d'automatisation renforcée sur toute la chaîne des processus de test.

Julien Botella

En tant que responsable de l'équipe Gravity chez Smartesting, Julien est passionné par le test logiciel, par le développement et il cherche à apporter des solutions pour améliorer le quotidien des testeurs, tout particulièrement des testeurs fonctionnels. Dans une période où les techniques d'IA et le traitement de langage naturel sont en rapide progrès, leur utilisation lui semble indispensable pour améliorer les outils de tests.

Vision du test en 2030 par Julien Botella

Le métier du testeur est enfin reconnu à sa juste valeur. Les sociétés ont compris l'importance d'inclure le testeur dès la conception du logiciel, en discussion avec les PMs/POs. Qu'il soit intégré à l'équipe de développement ou dans une équipe QA, le testeur est en interaction constante avec les développeurs

Comme le développeur, qui travaille avec un IDE, le testeur dispose d'outils modernes (ITE —> Integrated Testing Environments). Cet ITE en ligne, synchronisé avec l'IDE du développeur, lui apporte une assistance quotidienne. Compréhension des exigences fonctionnelles, user stories, risques mais aussi usage en production de l'application à tester… et la couverture par les tests de tous ces éléments ainsi que du code. Des algorithmes intelligents et l'IA vont apporter des données factuelles et des indicateurs qui permettront au testeur de prendre des décisions éclairées.

Le testeur va se concentrer sur le « quoi » tester et comment le prioriser, des agents autonomes (ou « testeurs virtuels ») prendront le relai pour l'exécution des tests.

Cristiano Caetano

Cristiano est un expert en tests logiciels, avec deux décennies d'expertise dans le domaine. Au cours des dix dernières années, son rôle a été essentiel pour guider les entreprises de test à construire et à lancer des outils de test innovants sur le marché. Actuellement, il occupe le poste de "Head of Growth" chez Smartesting, un éditeur qui se consacre au développement d'outils de test augmentés par l'IA.

Vision du test en 2030 par Cristiano Caetano

Au cours des dernières décennies, le secteur des tests a connu d'importantes transformations. D'abord négligé en tant que profession, il est aujourd'hui devenu une composante essentielle de toute équipe de développement. On est passé des tests manuels aux processus automatisés et, plus récemment, à l'implémentation de méthodologies de tests en continu.

Aujourd'hui, nous assistons à une nouvelle mutation qui se déroule sous nos yeux. L'adoption généralisée de l'IA pour soutenir les activités de test offre de nouvelles opportunités aux équipes de test pour améliorer leur productivité et fournir des logiciels de meilleure qualité à un rythme plus rapide.

J'entrevois un avenir où les équipes pourront effectuer des tests plus efficaces avec l'aide de copilotes alimentés par l'IA, améliorant ainsi leurs capacités de test. Elles pourront ainsi consacrer plus de temps à collaborer avec les membres de l'équipe pour remanier les exigences, explorer diverses approches de test au-delà des tests fonctionnels et s'attaquer à des scénarios complexes qui exigent une réflexion analytique approfondie et de la créativité par le biais de tests manuels.

Xavier Blanc

Xavier est professeur en Génie Logiciel à l'Université de Bordeaux et directeur du LaBRI (Laboratoire Bordelais de Recherche en Informatique). Ses recherches portent sur le développement et le test des applications web. Il est l'un des membres fondateurs de la société ProMyze, créée en 2016, qui adresse le domaine de la qualité des logiciels. Il travaille également pour les sociétés Smartesting et Mr Suricate, pour le développement de l'IA pour les tests logiciels.

Alexandre Vernotte

Alexandre a travaillé sur la cybersécurité des systèmes complexes à l'institut KTH en Suède. De retour à l'Université de Franche-Comté, il s'est intéressé à l'IA pour les tests, notamment leur priorisation. Il est maintenant ingénieur IA chez Smartesting, où il intègre l'IA aux produits de l'entreprise.

Vision du test en 2030 par Xavier Blanc et Alexandre Vernotte

Les récents résultats obtenus par l'IA et plus particulièrement par les IA génératives dans le domaine du développement logiciel annoncent un changement de paradigme important. La génération automatique de code est la première partie visible de ce changement. On voit aujourd'hui de nombreux outils de génération automatique de code accompagner les développeurs et leur permettre d'augmenter leur qualité et leur productivité. D'autres avancées sont encore à l'état de recherche mais des premiers résultats arrivent, notamment sur la planification et l'exécution automatique de tâches. Ces changements ont un impact positif sur le domaine du test logiciel. On comprend en effet aisément l'intérêt de la génération automatique pour les scénarios de tests et on mesure les apports des agents intelligents pour l'automatisation de l'exécution des tests. Un des défis majeurs qu'il reste à adresser pour les prochaines années réside dans le test des systèmes logiciels qui embarquent de l'IA. L'intégration de composants IA dans les systèmes logiciels commence à peine et pose déjà de sérieux problèmes, que cela soit dans la définition des scénarios, la mise en place des oracles ou la mise en œuvre d'environnements de test stables. C'est là, à notre humble avis, que la communauté du test a un rôle à tenir.

Sarah Leroy

Diplômée de l'ENSAI, filière Data Science & Ingénierie des données, Sarah Leroy est ingénieure Développement et Test chez Kereval, dans l'équipe Intelligence Artificielle (IA). Après un stage de fin d'études consacré à l'explicabilité de l'IA, elle continue à travailler sur des sujets autour du test de l'IA mais aussi de l'utilisation de l'IA pour le test. Elle a notamment animé l'atelier technique « Entraîner, évaluer et tester une Intelligence Artificielle fiable » lors de la JFTL 2023.

Eliott Paillard

Titulaire d'une Licence de Physique et d'un Master en Statistiques appliquées, Eliott Paillard consacre son stage de fin d'études à l'éthique dans l'Intelligence Artificielle (IA). Ingénieur d'études et de tests chez Kereval depuis 2023, il apporte son expertise sur des problématiques liées à la e-santé, aux smart cities et au test de l'IA.

Anne-Laure Wozniak

Ingénieure de recherche à Kereval depuis 2020, Anne-Laure termine actuellement son doctorat sur le test logiciel de la robustesse des systèmes utilisant de l'Intelligence Artificielle, en collaboration avec le Lab-STICC. Au cours de ses travaux, elle a notamment participé au programme national Confiance.ai afin de développer un outil de test de robustesse en boite noire pour les systèmes de vision par ordinateur.

Vision du test en 2030 par Sarah Leroy, Eliott Paillard et Anne-Laure Wozniak

Depuis plusieurs années déjà, il s'opère un changement de paradigme dans le développement, l'intégration et le déploiement des logiciels, notamment au travers du DevOps. Ces changements doivent être soutenus par des innovations dans le domaine du test logiciel. En particulier, l'automatisation des tests est une problématique centrale afin de gagner en efficacité et en fiabilité. Les avancées techniques dans le domaine de l'Intelligence Artificielle (IA) peuvent permettre d'apporter des réponses concrètes à ce besoin d'automatisation, permettant ainsi aux testeurs de consacrer davantage de temps à des tâches à plus haute valeur ajoutée. En revanche, l'émergence des systèmes basés sur l'IA nécessite de repenser les méthodes usuelles de test logiciel, voire de développer de nouvelles techniques pour s'adapter à leurs spécificités. De même, l'accélération de la génération automatique de code par l'IA va demander d'intensifier les tests logiciels. Ces thématiques seront donc au cœur des avancées dans les prochaines années.

La qualité du logiciel est une dimension essentielle de la valeur délivrée par les équipes IT. Cela donne une responsabilité essentielle aux professionnels des tests qui, en même temps, sont soumis à des évolutions très fortes dans les pratiques, les techniques et l'organisation des tests. Dans ce livre, le troisième édité par le CFTL, nous traitons de ces évolutions sous plusieurs angles : les défis et méthodes d'une qualité durable, l'arrivée de l'IA pour assister nos activités mais aussi comme objet de test et, enfin, au travers de différents éclairages sur des enjeux ascendants, tels que l'accessibilité, les pratiques d'ingénierie de la qualité et la cybersécurité.

Ce livre a pour but de permettre à chacun d'appréhender ces nouveaux défis, au niveau de ses pratiques mais aussi de son organisation. Comment intégrer la sobriété dans nos logiciels et nos pratiques de test ? Comment l'IA remet-elle en question nos processus et méthodes de test ? Comment appréhender la multiplicité des dimensions de la qualité IT ? Autant de questions abordées dans ce livre qui propose connaissances et axes de réflexion.

Ce livre est organisé en 4 parties.
En partie I, nous proposons un tour d'horizon de la qualité durable, c'est-à-dire de la prise en compte des enjeux écologiques dans les systèmes et dans les tests.
En partie II, nous parcourons les thèmes prégnants des enjeux actuels, tels que les tests dans le cloud, l'outillage des tests, l'accessibilité et la cybersécurité.
La partie III est consacrée à l'IA : pour assister les activités de test (avec quelles techniques et bonnes pratiques) et comme objet de test.
Enfin, la partie IV donne une vision du Testing à l'horizon 2030, apportée par les auteurs du livre, tous praticiens expérimentés des tests logiciels.

CFTL
Comité Français des Tests Logiciels

ISBN 978-2-95674-902-8
9 782956 749028