



Mr Suricate

ALL YOU NEED IS TEST

Facilitez-vous les tests, passez à l'automatisation !





Mr Suricate

Mai 2022

01

TESTER, OUI, MAIS
POURQUOI ? p.02

03

COMMENT METTRE EN PLACE
L'AUTOMATISATION DES TESTS ? p.14

02

L'AUTOMATISATION, UN INVESTISSEMENT
POUR LA QUALITÉ p.08

04

L'IA AU SERVICE DE
L'AUTOMATISATION DES TESTS p.28

INTRODUCTION

Au fil des ans, le domaine du test s'est vu être transformé par l'essor de l'automatisation. Et ce n'est que le début. Aujourd'hui estimé à 20,7 milliards de dollars, **le marché de l'automatisation des tests pourrait atteindre 49,9 milliards de dollars en 2026 d'après les dernières estimations***, soit un taux de croissance annuel composé de plus de 19%. Les raisons de cet engouement ? Face aux tests manuels, les tests automatisés offrent aux entreprises des avantages non-négligeables : gain de temps, mise en production accélérée, réduction du time to market, économies, pour n'en citer que quelques-uns.

Et si l'automatisation des tests peut paraître complexe, la réalité est toute autre. Certes, cela pose des challenges autant techniques qu'organisationnels, mais la bonne nouvelle, c'est qu'ils sont totalement accessibles avec une bonne préparation en amont. **Qui plus est, il existe aujourd'hui des solutions simples, qui ne nécessitent pas ou très peu de compétences techniques et qui permettent de rendre l'automatisation accessible à tous.**

Mais par où commencer ? Quels tests automatiser ? Quel outil choisir ? Quel budget allouer ? Dans ce livre blanc, vous trouverez tout ce qu'il vous faut savoir pour aborder l'automatisation de vos tests fonctionnels plus sereinement.

****étude Research and Markets 2021***





Mr Suricate

01

TESTER, OUI, MAIS POURQUOI ?

Un test logiciel, c'est tout simplement ce qui permet de vérifier qu'un produit ou une application logicielle fait ce qu'il est censé faire.



Prenons pour exemple le vélo. Avant de vous lancer sur la route, vous allez contrôler que les freins fonctionnent correctement, pour éviter tout accident. Vous allez donc tester les freins. Dans le cadre du test logiciel, le vélo n'est autre que le produit ou l'application logicielle, et les freins correspondent à l'une de ses fonctionnalités. Si les freins ne répondent pas ou ne freinent pas bien, on peut alors comparer cela à une anomalie, ou un bug, dans le domaine du test.

Auparavant, le test logiciel était considéré comme un moyen de détecter des erreurs sur les sites web ou applications mobiles, mais aujourd'hui, cela va bien plus loin. En réalité, le test logiciel consiste à explorer une interface afin de vérifier et valider le bon fonctionnement de différents critères (fonctionnalités, sécurité, performance, ergonomie...), de repérer des comportements inattendus, des possibilités imprévues et, dans un second temps, de remonter les anomalies trouvées afin de les corriger au plus vite.



QUELS SONT LES DIFFÉRENTS TYPES DE TESTS?

1. LES TESTS FONCTIONNELS

Il s'agit d'une famille de test qui permet de vérifier qu'un logiciel (site web, application mobile, API...) fonctionne conformément aux spécifications déterminées par le client. Chaque fonctionnalité de la plateforme est ainsi testée pour contrôler qu'elle correspond à ses besoins.

Sur un site e-commerce, cela consiste, par exemple, à vérifier que la connexion au compte ou encore que l'ajout au panier ou la sélection d'un mode de livraison ne pose aucun problème et qu'il n'y a pas de différences constatées avec les données fournies au préalable.

Les tests fonctionnels se composent de plusieurs niveaux :



LES TESTS UNITAIRES

Ce sont les tests qui permettent à un développeur de vérifier son code au niveau du composant qu'il doit réaliser, en testant les fonctionnalités de façon individuelle.

Exemple : Cela peut consister à tester le bon fonctionnement du bouton « ajouter au panier » quand on clique dessus.



LES TESTS D'INTÉGRATION

Ce sont les tests qui permettent de s'assurer que les différents composants d'un logiciel fonctionnent bien ensemble et délivrent les résultats attendus.

Exemple : Cela peut consister à vérifier que le message envoyé après l'appui sur le bouton « ajouter au panier » est bien reçu par le serveur.



LES TESTS SYSTÈME

Ce sont les tests qui permettent de contrôler que l'ensemble des composants intégrés au système (le logiciel ou l'application) fonctionnent sur des parcours clients ciblés.

Exemple : Cela peut consister à vérifier que le panier fonctionne bien, que les bonnes erreurs sont remontées.



LES TESTS D'ACCEPTATION

Ce sont les tests effectués par le métier ou les utilisateurs finaux. Il s'agit souvent de parcours de bout en bout, visant à valider l'expérience utilisateur dans un contexte complet.

Exemple : Cela consiste à vérifier que le parcours fonctionne dans son ensemble, de la home page au paiement.

2. LES TESTS NON FONCTIONNELS

Il s'agit d'une famille de test qui permet de vérifier les aspects non fonctionnels d'un système. Cela peut notamment concerner la sécurité (protection des données, accès à l'application), la performance (temps d'exécution, débit), la charge (capacité maximale du nombre de visiteurs), l'utilisabilité ou encore l'accessibilité.



Dans le cadre d'un vélo, il s'agirait notamment de tester le confort de la selle, la vitesse maximale jusqu'à laquelle vous pouvez rouler, la facilité à pédaler, etc.

Les problèmes qui ne sont pas liés aux fonctionnalités peuvent tout autant impacter négativement l'expérience utilisateur. Il est donc important que l'application soit également testée pour les exigences non fonctionnelles.

3. LES TESTS DE NON RÉGRESSION

Les tests de non régression (TNR) consistent à vérifier que les modifications apportées sur le système (le logiciel ou l'application) n'ont pas impacté les fonctionnalités précédemment existantes.

Pour reprendre l'exemple du vélo, si on change la roue, un test de non régression consisterait à vérifier que malgré le changement de roue, les freins fonctionnent toujours (mais il n'est pas forcément nécessaire de vérifier que la pompe à vélo s'accroche bien).



4. LES TESTS EXPLORATOIRES

Il s'agit de tester son application en utilisant son expérience et ses connaissances. Le test exploratoire se base sur la découverte, l'exploration et l'investigation d'un produit à tous les niveaux : fonctionnel, graphique, ergonomique...

Par exemple, cela consisterait à faire essayer votre vélo à vos amis afin qu'ils vous fassent des retours propres à chacun, dans un cadre défini. Certains peuvent le trouver trop petit, trop grand, ou encore trouver que la selle n'est pas confortable ou que les freins sont trop serrés.

Le but, ici, est d'exécuter des cas de tests réels, qui n'ont pas été imaginés au préalable.

Dans certaines entreprises, des tests days sont organisés, afin que tout le monde puisse essayer l'application sans avoir de cadre ou de cahier de recette, permettant ainsi d'avoir des remontées sur les évolutions testées.



Que peut-on tester ?

Tout peut être testé, à partir du moment où c'est testable. On peut notamment tester des interfaces web (sites, applications, ERP...), des applications mobiles, des **API***, du code, des infrastructures, des architectures, etc.

Sur quels environnements teste-t-on ?

Les tests sont effectués sur des environnements. Il faut voir cela comme des plateformes qui permettent de tester le logiciel ou l'application avec des données fictives, sur lesquelles il est possible de faire de faux paiements, par exemple.

En d'autres termes, les environnements reproduisent le logiciel ou l'application afin de pouvoir effectuer les tests dans des conditions réelles.

Ils permettent de :

- Tester l'application dans des conditions spécifiques
- Ne pas impacter directement les utilisateurs
- Maîtriser les données

On distingue les environnements de test (recette et pré-production) et l'environnement de production. Dans le cas des environnements de test, il faut savoir qu'ils ne sont pas dimensionnés comme une production, pour des raisons évidentes de coût. À noter qu'il existe également des environnements de développement, qui permettent aux développeurs de livrer leur code et sur lesquels des tests peuvent également être réalisés.

**API = Application Programming Interface. Interface permettant d'interroger un logiciel via des requêtes web au lieu d'une interface graphique.*





Le Suri-tips :

Une bonne pratique est de tester partout et tout le temps, surtout dans les **méthodologies agile*** où il est important d'avoir un statut récurrent et surtout rapide de la qualité de son applicatif, jusqu'au monitoring de votre production. Le monitoring en production consiste à mettre en place des indicateurs et des alertes pour vérifier que tous les services et sous-services fonctionnent (y compris ceux de vos prestataires).

L'environnement de recette :

La recette est un environnement accessible aux équipes de tests pour réaliser les tests systèmes sur des données fictives, anonymisées, sans risque d'impacter la production. Cette plateforme contient en général les évolutions qui vont arriver prochainement en **production***.

L'environnement de pré-production :

C'est l'environnement qui est au plus proche de la production. Il comprend généralement des données de production et les évolutions qui sont prévues d'être **mises en production***. Cet environnement reproduit identiquement celui de la production, afin que les tests et vérifications réalisés soient les plus pertinents possibles.

À noter : certains environnements peuvent être créés pour faire des tests de charge, qu'une recette ne pourrait pas supporter.

L'environnement de production :

Il s'agit ici de la destination finale du logiciel ou de l'application, lorsque celui ou celle-ci est mis en service et donc disponible en ligne, pour une utilisation réelle. En d'autres termes, c'est l'environnement sur lequel l'utilisateur final navigue.

Le monitoring en production est important car même si votre site ou application est d'une excellente qualité, vous n'êtes pas à l'abri qu'un bug provienne de vos prestataires externes, comme par exemple au niveau du paiement.



Pour rester dans la métaphore du vélo, ce dernier peut être parfaitement bien testé, mais la piste cyclable que vous allez emprunter peut être semée d'imprévus. Est-elle bien entretenue ou pleine de nid-de-poule ? Va-t-elle être plus empruntée que d'habitude et donc plus dangereuse ou plus lente ? Est-elle bien sécurisée et séparée du flux de voitures par de bonnes infrastructures ? Faire du monitoring en production, c'est analyser toutes ces données.

Quel est l'intérêt du test pour les équipes métiers* ?

Le succès de votre projet web ou mobile repose avant tout sur la satisfaction des utilisateurs. Et aujourd'hui, ces derniers exigent des expériences fluides. Il suffit d'un seul bug pour impacter négativement votre entreprise, que ce soit au niveau de son image mais aussi au niveau de son chiffre d'affaires.

Selon une enquête réalisée par Mr Suricate auprès de 14 800 internautes, **86 % des Français avouent notamment qu'ils renoncent à un achat à cause d'un bug et 70 % vont acheter sur un autre site e-commerce**. Et quand ils n'abandonnent pas, les utilisateurs peuvent également laisser des commentaires négatifs.

Le constat est donc sans appel : il faut faire la chasse aux bugs pour offrir l'expérience utilisateur la plus satisfaisante possible et préserver votre image de marque. Et c'est en multipliant les tests que vous y arriverez. Faire des tests, c'est prendre la main sur la qualité de l'expérience utilisateur (UX).

Mais les tests sont également importants car ils permettent d'identifier les bugs plus tôt et de les corriger ainsi plus rapidement. Les bénéfices sont doubles : cela permet non seulement de réduire les coûts de résolution de ces anomalies, mais cela permet également de garantir la fiabilité, la sécurité et la performance du produit, ce qui ne fera qu'en augmenter sa qualité.

**Agile : Méthodologie de gestion de projet qui prône la collaboration entre des équipes auto-organisées et pluridisciplinaires avec leurs clients.*

**Production : Cadre dans lequel les logiciels, sites web, applications mobiles, sont réellement mis en service pour les utilisateurs finaux.*

**Mise en production : Déploiement d'une nouvelle version d'un logiciel/d'un site/d'une application sur un environnement de production.*

**Équipes métiers : responsable marketing, chef de projets digitaux, directeur e-commerce, directeur digital, responsable e-commerce, chef de produit, responsable UX, responsable des achats, etc.*



L'AUTOMATISATION, UN INVESTISSEMENT POUR LA QUALITÉ

Qu'est-ce que l'automatisation des tests ?

L'automatisation des tests consiste à utiliser un outil qui va reproduire les actions/vérifications de test, à la base réalisées manuellement par un être humain, en les répétant de manière automatisée à l'infini.

Dans le cas des tests automatisés, ce sont des programmes qui exécutent les tests, tandis que dans le cas des tests manuels, ces derniers sont réalisés par des humains (testeurs, développeurs, chefs de projet...).

Mais dans les deux méthodes, l'idée est la même : se mettre à la place des utilisateurs finaux pour effectuer chaque action et vérifier que tout se comporte conformément aux résultats attendus. L'objectif étant de réduire les temps d'exécution des tests manuels en automatisant notamment les tâches répétitives ou complexes à réaliser manuellement.

POURQUOI AUTOMATISER ?

Améliorer le move to prod et réduire le time to market



L'automatisation permet d'exécuter les tests plus rapidement, si bien que vous pouvez vous rendre compte plus vite de la qualité de votre application et accélérer ainsi sa mise en production. Vous allez ainsi industrialiser toutes vos démarches de tests pour les rendre moins fastidieuses, plus rapides et plus efficaces avec moins d'erreurs humaines.

Et si vous partez plus vite en production, vous réduisez le **Time To Market***. C'est un atout important pour se démarquer de ses concurrents en déployant plus rapidement des fonctionnalités attendues.

Gagner en temps de tests

Certaines vérifications peuvent prendre plusieurs heures quand elles sont faites manuellement, alors qu'un automate peut les faire en quelques minutes quand on permet de réaliser des exécutions par centaine en simultané.

Pour exemple, chez Mr Suricate, en 2021, nous avons réalisé 10 741 424 exécutions de scénarios, avec une moyenne de 3,7 minutes par exécution. Cela équivaut à 27 600 jours, soit 75 ans de tests manuels économisés à tous nos clients confondus sur l'année.



En automatisant certains tests, notamment les plus récurrents et répétitifs, les équipes techniques n'ont plus qu'à se charger de l'analyse et de la correction des incidents, ils peuvent donc libérer du temps pour d'autres missions à plus forte valeur ajoutée, comme la validation des prochaines évolutions, l'exécution de tests manuels plus complexes ou encore la mise en place de nouveaux tests automatisés.

Décharger les équipes techniques



Garantir une couverture constante

L'inconvénient des tests manuels est qu'ils ne permettent pas de vérifier toutes les fonctionnalités à chaque recette, uniquement les plus critiques, auquel cas cela prendrait beaucoup trop de temps. Grâce à l'automatisation, il est possible de mettre en place des tests à chaque correction, mise à jour, déploiement. Et il est également possible de les exécuter au plus tôt dans le processus, de la conception du projet à sa mise en production, et même au-delà. Cela permet ainsi de garantir une couverture constante des fonctionnalités.



Faut-il tout automatiser ?

Il pourrait être tentant d'automatiser la totalité de ses tests, n'est-ce pas ? Mais ce serait une erreur. Et pour cause, bien que plutôt aboutis, **les tests automatisés ne se suffisent pas à eux-mêmes pour apporter une évaluation complète et pertinente d'une application**. Et surtout, il n'est pas possible de tout automatiser. Par exemple, les tests exploratoires, de par leur côté informel, sont difficilement automatisables sans recours à l'intelligence et le ressenti humain.

Qui plus est, certains tests ont besoin de rester manuels, car le fait qu'ils soient exécutés par un humain permet de repérer des choses qu'une machine ne verrait pas, notamment en ce qui concerne l'ergonomie, **l'accessibilité*** etc.

Il est donc nécessaire de combiner les tests manuels et automatisés pour assurer la qualité des développements des applications web ou mobiles et obtenir ainsi un retour sur investissement optimal.

Par exemple ...



Il est possible de motoriser son vélo pour qu'il roule tout seul, mais si l'humain ne dirige pas le vélo, sa trajectoire restera difficilement droite et il n'ira pas forcément dans la bonne direction.



**L'accessibilité numérique consiste à rendre les services en ligne accessibles aux personnes en situation de handicap.*



COMMENT CHOISIR LES TESTS À AUTOMATISER ?

Bien choisir les tests que vous allez automatiser est d'autant plus important qu'un mauvais ciblage peut rendre votre stratégie d'automatisation totalement inefficace et vous faire perdre plus de temps et d'argent qu'autre chose.

Il y a deux premières questions à se poser avant d'aller plus loin :



Quels sont les objectifs de votre démarche d'automatisation ?



Quels sont les moyens que vous pouvez mettre en œuvre ?

Pour savoir maintenant quels sont les critères d'automatisation à considérer en priorité, nous vous conseillons de commencer par :



La répétitivité

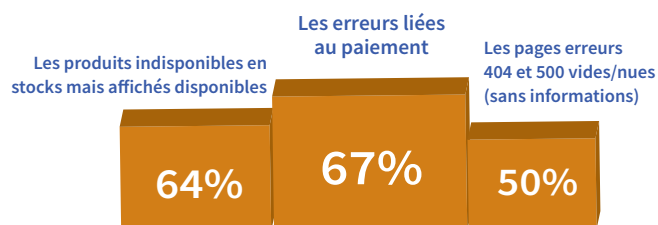
Quels tests effectuez-vous à chaque déploiement ? C'est en automatisant les tests les plus récurrents et les plus répétitifs (tests de non régression, tests d'acceptation...) que vous pourrez obtenir un gain de temps considérable et ainsi vous consacrer à l'exécution de tests plus complexes.



La criticité des parcours

Il est pertinent de cibler les tests fonctionnels qui concernent les parcours et les fonctionnalités critiques, c'est-à-dire ceux qui ont un enjeu direct pour l'entreprise (financier, juridique, image). Par exemple, sur un site e-commerce, on conseille d'automatiser les tests qui portent sur le tunnel d'achat car ils sont générateurs de chiffre d'affaires et le moindre bug sur ces parcours est impactant financièrement. Souvent le coût de l'automatisation des tests est amorti au premier bug évité ou trouvé sur le parcours d'achat car cela aura empêché une perte de chiffre d'affaires.

Il faut savoir que, d'après une étude de Mr Suricate, les trois bugs qui énervent le plus les internautes sont :



La multi canalité

Mener des tests sur plusieurs navigateurs (Chrome, Firefox, Safari) ou sur plusieurs devices (smartphone, tablette, ordinateur) peut vite faire monter le nombre d'exécutions à réaliser et c'est notamment lorsque le volume de tests s'amplifie que l'automatisation devient pertinente.



La performance

Plus les problèmes de performance sont détectés tard, plus ils sont coûteux à corriger. Il est donc logique d'automatiser ces tests de performance et de charge, de façon à pouvoir en repérer les ralentissements au plus vite.

EXEMPLES PAR TYPES DE SCÉNARIOS



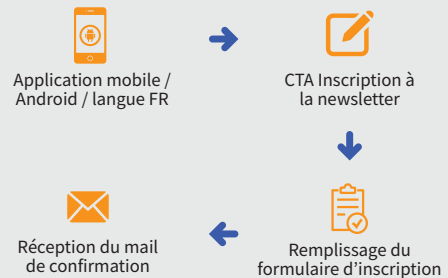
PARCOURS DE BOUT EN BOUT

Orienté sur un parcours généraliste proche de ce que pourrait réaliser l'utilisateur de l'application.



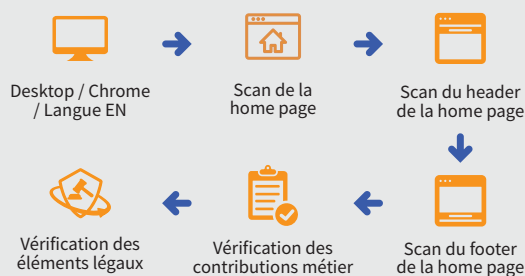
TEST DE FONCTIONNALITÉ

Plus court et ciblé sur les fonctionnalités moins critiques : compte client, store locator, newsletters ...



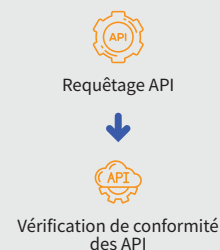
TEST DE COMPARAISON GRAPHIQUE

Orienté luxe et multi-pays, les tests de comparaison graphique permettent de vérifier que le site ou l'application mobile ressemble à sa version précédente en mode "pixel perfect" et qu'il y a une conformité légale/juridique.



TEST D'API

Il permet de vérifier le monitoring en production, la rapidité du requêtage API et la conformité des réponses de l'API.



COMMENT METTRE EN PLACE L'AUTOMATISATION DES TESTS ?

L'automatisation des tests est un investissement sur le long terme car cela a un certain coût. Problème : beaucoup sont les projets d'automatisation de tests qui n'aboutissent pas au résultat escompté. La raison ? Le plus souvent, c'est parce qu'ils n'ont pas été assez réfléchis en amont.

Pour réussir son automatisation et bénéficier de ses nombreux avantages, il est impératif de mettre en place une stratégie adaptée, non seulement à son contexte, mais aussi à ses besoins et ses contraintes. Cette stratégie doit être partagée, co-construite et validée par tous les acteurs du projet (développeurs, testeurs, métier).

Et surtout, elle doit prendre en compte tous les aspects fonctionnels (ou non) afin d'implémenter les bons types de tests au bon endroit, du code à l'interface.

Mais par où commencer ?

03







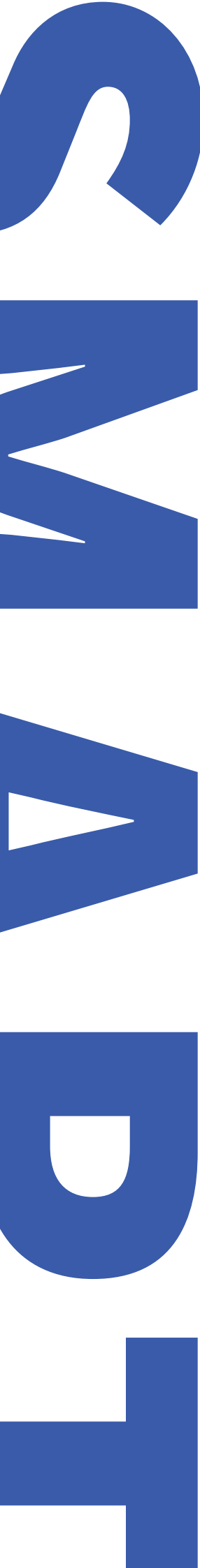
Fixer ses objectifs

Pour déterminer les tests que vous voulez automatiser, il faut avant tout déterminer POURQUOI vous voulez les automatiser. Mais entre les objectifs qu'on voudrait réaliser et ceux qui sont possibles, il peut y avoir un écart important. Il convient donc d'adapter vos objectifs aux ressources financières et humaines et au temps dont vous disposez. Le but est de savoir si ces derniers sont pertinents, réalistes et surtout faisables par rapport à votre situation.

Les objectifs de l'automatisation des tests peuvent être de plusieurs ordres, mais voici quelques exemples pour vous orienter :

**Chaîne d'intégration continue : une méthode de développement ayant pour objectif de corriger plus rapidement les bugs, d'améliorer la qualité du code et de réduire le temps nécessaire pour valider et publier de nouvelles mises à jour du logiciel.*





Pour fixer vos objectifs efficacement, utilisez la méthode SMART :

Spécifique

Précisez ce que vous voulez faire

*Ex : Implémenter les tests dans une chaîne d'intégration continue**

Mesurable

Utilisez des indicateurs précis

Ex : Gagner du temps sur l'exécution des campagnes de test

Ex : Diminuer les coûts de correction en détectant les anomalies plus tôt

Atteignable

Définissez des objectifs réalisables, dans votre champ d'action

Ex : Économiser de l'argent sur l'exécution des campagnes de test

Ex : Améliorer la qualité de l'application en évitant les bugs critiques

Réaliste

Fixez des objectifs qui correspondent à vos valeurs et à votre stratégie

Ex : Assurer une meilleure couverture de test

Temporel

Soyez précis sur le délai

Ex : Augmenter le taux de conversion par 5 en 1 an



Rendre l'application testable et automatisable

L'automatisation des tests n'est pas complexe à mettre en place, sauf si l'application, l'API, le code ou l'interface en question s'avère difficile à tester. Pour éviter cela et faciliter ainsi la possibilité d'automatiser ses tests, il convient de s'assurer d'avoir de la donnée assez conséquente pour supporter un nombre d'exécution important, mais également de mettre en place des marqueurs qui permettent de cibler un élément ou de remplir des champs plus facilement.

Dans le cas des marqueurs, il faut savoir qu'ils ne sont pas compliqués à mettre en place par les développeurs. Il y a une réelle réflexion à avoir sur le sujet dans les équipes de développement, car en plus de rendre l'automatisation des tests plus simple, cela permet aussi aux développeurs de cibler les éléments utilisés par l'automatisation et ainsi être plus prévenants et proactifs en cas de changement auprès des testeurs.

Les données de test correspondent à toutes les informations qui sont nécessaires pour exécuter correctement les tests. Il peut s'agir d'informations de contact, de listes de produits, de prix, de numéros de sécurité sociale, des informations de cartes de crédit, etc. Elles doivent être précises et complètes afin d'optimiser la détection des bugs. Il en va de la qualité des tests et donc, par conséquent, de l'application.



Le Suri-tips :

D'après la dernière enquête du CFTL*, près de 57% des répondants déclarent ne pas avoir d'outil de gestion des données de tests, ce qui peut considérablement complexifier l'automatisation des tests. Il convient de mettre en place une véritable stratégie les concernant car le succès de l'automatisation des tests repose notamment sur la bonne gestion des données de tests.

**Enquête CFTL 2021-2022 sur les pratiques des tests logiciels*



Choisir le bon outil

Pour répondre au besoin de l'automatisation, de nombreux outils existent, mais il ne faut pas se tromper. Tous les outils ne se valent pas. Certains peuvent être trop techniques et demander des compétences qui ne sont pas adaptées à celles de votre équipe. D'autres ont des coûts d'implémentation et de maintenance élevés et il faut donc le prendre en compte. Et d'autres ne permettent pas de tout tester.

Cela demande donc de bien étudier vos besoins, de prendre en compte votre projet de développement (s'il s'agit d'une API, d'un client lourd, d'une interface web, d'une application mobile, si c'est multi-devices...), les compétences de votre équipe, le niveau de complexité des tests fonctionnels que vous souhaitez automatiser, ainsi que le coût que vous souhaitez y allouer. Un autre point à prendre compte : toute l'infrastructure qui permet d'exécuter les tests derrière. Certains le proposent, mais dans d'autres cas, il faut mettre en place une architecture complexe qui nécessite des compétences différentes, telles qu'un administrateur système ou un architecte (aussi appelé OPS).

Trois solutions pour automatiser ses tests

EN INTERNE, DANS L'ÉQUIPE



COMPÉTENCES

Compétences techniques disponibles ou recrutement de profils techniques à prévoir.



MISE EN OEUVRE

Écriture de code pour les développeurs, infrastructure architecturée à mettre en place, reporting à créer.



COUVERTURE

Experts et outils différents à trouver pour chaque type de test.



MAINTENANCE

Infrastructure à maintenir, tests automatisés techniques à documenter et maintenir.

EN EXTERNE, AVEC MR SURICATE



COMPÉTENCES

Délégation des compétences techniques à une équipe d'experts.



MISE EN OEUVRE

Pas de gestion d'infrastructure, technologie no-code et sans intégration, plateforme de reporting SaaS, conception de stratégie de tests.



COUVERTURE

Complète (tests fonctionnels, graphiques, d'API, de performance, d'accessibilité, SEO).



MAINTENANCE

Un chargé de recette maintient jusqu'à 400 scénarios par mois, analyse et remontée des anomalies.

EN INTERNE, AVEC LA SOLUTION NO-CODE / CODELESS



COMPÉTENCES

En quelques jours de formation sur un outil no-code, un utilisateur novice va pouvoir réaliser le même travail qu'un développeur de tests automatisés.



MISE EN OEUVRE

Accélération de la mise en œuvre en s'affranchissant du choix d'un outil technique, du recrutement d'experts et de toute l'infrastructure technique.



COUVERTURE

Les solutions codeless couvrent de plus en plus de périmètre, mais peu propose chaque type de tests, ce qui nécessite d'avoir plusieurs outils.



MAINTENANCE

Toute l'infrastructure si installée en local, sinon uniquement les scénarios lorsque le site est modifié et les scénarios en échec.





Bien analyser

Une bonne stratégie d'automatisation repose sur une analyse rapide et efficace des scénarios de tests automatisés tombés en échec.

Après tout, quel intérêt de lancer des campagnes de tests automatisés si vous décidez de livrer en production sans contrôler que des anomalies ont été détectées ? Le risque est alors de déployer une application de mauvaise qualité, remplie de bugs critiques et on ne parle pas de l'impact négatif que cela peut avoir sur vos utilisateurs.

Exécuter des tests automatisés par dizaine, centaine ou milliers ne suffit pas, il faut également mettre en place des systèmes d'analyse des échecs et en faciliter la remontée, en créant, par exemple, automatiquement des fiches d'incidents, afin de pouvoir prendre les actions nécessaires par la suite : relancer le test qui est tombé en échec, le mettre à jour ou corriger l'anomalie détectée.



Le Suri-tips :

Pensez aux logiciels de bug tracking comme Jira®, Mantis® ou encore Redmine®, qui vous permettent de classer vos incidents via des tickets et de les centraliser, et mieux, à intégrer votre solution d'automatisation à ces outils pour avoir une création automatique d'incident à analyser.



Le Suri-tips :

Faites en sorte que les scénarios de tests soient simples à maintenir en documentant et en rendant le code lisible. Pour cela, mettez en place des messages d'erreurs compréhensibles et des traces d'erreurs propres et claires.

Bien maintenir

« Un grand pouvoir implique de grandes responsabilités » et dans le cas de l'automatisation, les responsabilités concernent surtout la maintenance des tests. Il faut savoir que l'automatisation échoue souvent sur le long terme à cause d'un manque de maintenance sur un trop grand nombre de tests (c'est pour cela qu'il faut être réaliste sur le nombre de tests que vous automatiserez). Il convient donc de mettre à jour ses scénarios de tests automatisés dès qu'il y a des modifications, que ce soit au niveau de **l'UI***, des fonctionnalités, du système d'exploitation, du navigateur, des données, etc.

Une maintenance efficace consiste également à analyser les tests automatisés qui tombent en échec. Pourquoi ? Parce que si cela peut être dû à un véritable bug, cela peut également être dû à une erreur dans le code ou dans l'exécution des tests et pouvoir le détecter permet ainsi de pouvoir régler le problème.

Si la maintenance des tests automatisés représente un certain coût, faire l'impasse dessus serait une erreur car le risque, ici, est de ne pas bien détecter les anomalies et de faire ainsi perdre du temps aux équipes, alors que le but même de l'automatisation est de leur en faire gagner.



**UI : Interface utilisateur. Ensemble des éléments graphiques et textuels qui permettent une interaction entre l'utilisateur et le site internet, l'application ou le logiciel.*

Mettre en place des indicateurs, et les suivre !

Pour évaluer si l'automatisation de vos tests fonctionne et répond bien aux objectifs que vous vous êtes fixés, il convient d'instaurer un suivi en mettant en place un tableau de bord avec des données et métriques personnalisées.

Cela permet de voir comment progressent les exécutions de scénarios dans le temps, mais aussi de se rendre compte du nombre de succès et d'échecs, ou encore de savoir combien il y a d'incidents à l'instant T.

Et surtout, cela permet de piloter plus facilement et plus rapidement ses tests afin d'adapter sa stratégie au fur et à mesure. Sans cette visibilité, on perd un gain considérable de l'automatisation.

Quels indicateurs clés de performance (KPIs) mettre en place ?*



Le dénombrement des anomalies : le pourcentage de bugs détectés avant déploiement, le nombre ou pourcentage de bugs majeurs/critiques en production, le nombre de bugs détectés par test...



La satisfaction des utilisateurs : notes, commentaires, avis...



L'analyse des anomalies : le pourcentage des régressions dans les bugs en production, le pourcentage des anomalies liées à des composants particuliers, le pourcentage des anomalies liées à des critères non fonctionnels...



Les anomalies non fonctionnelles : le temps de réponse avec un nombre d'utilisateurs défini (charge), le nombre de warnings de sécurité majeurs ou critiques, le temps de réponse conforme sur les requêtes...



Les tests en développement : le pourcentage de tests en échec lors d'une validation...



Les couvertures : le pourcentage de couverture des fonctionnalités sur une campagne...

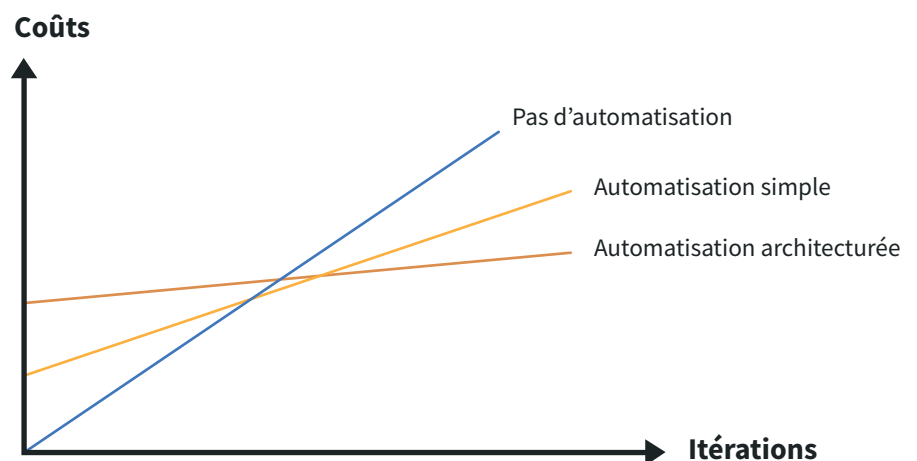


Le Suri-tips :

Attention, il ne faut pas confondre objectif et métrique. Une métrique ne doit pas être un objectif, mais uniquement un moyen d'évaluer ses objectifs. Obtenir l'aide d'experts peut vous aider dans la sélection des KPIS les plus pertinents pour votre démarche d'automatisation des tests.

Mais combien ça coûte, tout ça ?

Coût d'écriture, coût d'exécution, coût de maintenance, coût de formation, coût des infrastructures, coût des licences... À première vue, l'automatisation des tests semble coûter cher. En réalité, tout dépend de l'entreprise, de ses ressources, de son projet web (ou mobile), de la solution d'automatisation choisie, etc. Et s'il est vrai que sa mise en place représente un certain budget au départ, le ROI (retour sur investissement) de l'automatisation n'est pas à démontrer sur le long terme.



Si nous ne pouvons vous donner une estimation spécifique du coût de l'automatisation, voici un exemple de retour d'expérience d'un client de Mr Suricate issu du secteur du e-commerce alimentaire, afin que vous puissiez vous rendre compte du retour sur investissement que peut générer l'automatisation des tests :

ROI - AUTOMATISATION DES TESTS

E-commerce alimentaire



SANS MR SURICATE



✓ 20 à 50 50 à 100 + de 100

Automatisés ✓ **Manuels**
✓ **Externalisés** (75%)

✓ Recette ✓ Production



**25-30K€
PAR MOIS**

Ressources internes .
Coûts externes .
Revenus perdus en raison .
d'instabilités .

AVEC MR SURICATE



✓ 20 à 50 50 à 100 + de 100

 **Mr Suricate**

✓ Recette ✓ Production



**2-7K€
PAR MOIS**

. Gain de temps (Qualification des bugs /
Mise à jour des scénarios)
. Économies (18-25K€) (Automatisation de
tests manuels / Tests récurrents en recette)
. Revenus et fidélisation client (Tests
récurrents en production / Proactivité et
réactivité)

PARCOURS CLIENT

TESTS

ENVIRONNEMENT

VOLUME DES BUGS
CONSTATES

COÛTS

04

L'IA AU SERVICE DE L'AUTOMATISATION DES TESTS

Depuis quelques années, les innovations en matière d'apprentissage automatique et d'intelligence artificielle contribuent à faire évoluer de nombreux secteurs d'activité. Et le monde du test en fait partie. Mais comment l'IA peut-elle s'inscrire dans l'automatisation des tests ? Comment peut-elle aider les équipes métiers à mieux appréhender ce domaine ? Quels sont ses gains, ses risques et ses limites ? Nous avons interviewé trois experts pour répondre à ces questions.

Interview Marc Hage Chahine



« L'IA devrait être un outil d'aide à la décision, mais l'IA ne devrait pas prendre la décision. »

Expert méthodes et outils chez Sogeti, Marc Hage Chahine a toujours travaillé dans le test et souhaité partager ses expériences. Il a la chance de pouvoir le faire dans son travail quotidien, mais aussi à travers l'écriture d'articles sur le blog de la Taverne du testeur ou des magazines comme Programmez!. Marc Hage Chahine a également le plaisir de participer à l'organisation de grands événements du test comme la STLS (à Sophia Antipolis) et la JFTL.

L'automatisation des tests est un marché en pleine évolution. D'après les dernières estimations de Research and Markets (2021), ses revenus pourraient atteindre les 49 milliards de dollars en 2026.*

À votre avis, quel avenir peut-on attendre pour l'automatisation des tests ?

L'automatisation des tests est censée être le futur proche depuis plus de vingt ans. On n'arrête pas d'en parler comme un remplacement du test manuel. Mais on voit bien que ça ne remplace pas

le test manuel et que ça ne le remplacera pas, mais aussi que son implémentation est complexe. Si on regarde les dernières enquêtes du CFTL, l'automatisation des tests n'avance pas sur les deux dernières années, alors qu'on est dans des contextes agiles où l'automatisation est toujours plus nécessaire.

Pourquoi ça n'avance pas ?

Pendant longtemps, l'automatisation est restée très technique, avec du code, du développement. Mais depuis un certain nombre d'années, on commence à voir des outils sans code ou sans script, qui ont émergé avec le **Keyword Driven Testing*** dont le principal représentant est Robot Framework®. C'est la génération zéro du test pour les non-techniques. On a Gherkin® qui va faire de l'automatisation sur le même principe. On a d'autres outils qui commencent à être de plus en plus matures, plus accessibles et fonctionnels. Mais même s'ils sont accessibles, cela reste des outils assez techniques et pensés à partir du code, du script.

Depuis, de nouveaux outils sont en train de sortir : Keysight® avec Eggplant, Tosca® de Tricentis ou la dernière version d'UFT®, où ils essaient d'ajouter de l'IA, des blocs, des modules, pour que les gens puissent les utiliser plus facilement. Il y a aussi beaucoup de reconnaissance d'images, ce qui permet d'être agnostique au code et d'avoir un seul script pour faire tourner son test sur différents téléphones (iOS, Android), sur des tablettes, des PC voire même de l'embarqué. La prochaine évolution sera l'IA. On voit des émergences avec des outils comme Gravity®, de Smartesting, qui va analyser les logs pour proposer des parcours et les automatiser.

Attention, ce n'est pas parce qu'il y a tous ces outils qu'ils vont tous être utilisés. Il y a des barrières de prix, est-ce qu'il faut faire cet investissement ou pas. Il faut aussi que les équipes qui sont en charge du développement des produits aient la bonne politique, la bonne stratégie au niveau des tests et savoir gérer la qualité. Il ne faut pas l'oublier : pour avoir des bons tests automatisés, il faut avant tout avoir des bons tests. Cette nécessité de l'automatisation doit se faire ressentir par les équipes, l'automatisation qui ne doit pas reposer uniquement sur le testeur.

En parlant de l'IA, comment la voyez-vous s'inscrire dans l'automatisation des tests ?

Pour moi, l'IA est là pour faciliter la décision, les actions. Ce n'est rien de plus que de l'automatisation plus poussée que juste de l'automatisation de l'exécution. Cela doit aider à faire des choix, mais l'IA ne fera pas tout. Il faut des données, des informations et donc il faut des personnes derrière. Avec l'IA, on arrive à battre des champions de go ou d'échecs, mais l'IA des échecs ne va pas battre un champion de go. Elle est très spécifique et c'est donc sur un cas spécifique qu'elle sera plus performante que nous et nous permettra d'avancer. Il y a beaucoup de cas d'utilisation avec l'IA, notamment sur la priorisation des tests, qui va déboucher sur de la sélection. Par exemple, on a 500 tests de régression, lesquels faut-il faire en premier pour trouver le plus de bugs ? Mais au final on va pouvoir n'en faire que 400 ou 450, donc il faut déterminer lesquels on ne va pas faire.

Quels gains peut apporter l'IA à l'automatisation des tests ?

L'IA peut aider sur plusieurs points : elle peut aider sur l'analyse, notamment des tests en échec, elle peut aider sur les cas d'utilisation mis en avant par les éditeurs, qui sont en échec, pourquoi, et même les faire refunctionaliser et corriger les scripts directement. L'IA va aussi aider sur la maintenance et l'entretien en permettant de mieux gérer le répertoire de tests. Par exemple, si des tests sont en doublons, l'IA va nous dire qu'il faudrait penser à les supprimer car ils n'ont pas d'intérêt, ou bien elle va nous dire de retravailler les données de certains tests, etc. Et l'IA va aussi aider à la construction : on dit ce qu'on veut faire et le script se fait derrière, c'est d'ailleurs ce que font déjà les outils sans script.

La seule vraie limite de l'IA ça va être ce que nous, on imagine en faire. Les autres limites vont être les coûts d'implémentation et également l'intérêt de le faire, car ce n'est pas parce qu'on a imaginé faire quelque chose que c'est vraiment intéressant. Par exemple, les voitures volantes, ça fait 50 ans qu'on les imagine et qu'on pourrait très bien les faire mais au final ça ne sert à rien, ça n'apporte pas de plus-value et ça consomme trop d'énergie. Il faut qu'il y ait un besoin.

**Keyword Driven Testing : Tests basés sur des mots-clés. Méthodologie qui utilise des mots-clés (ou mots d'action) pour symboliser des actions à réaliser pour tester une fonctionnalité.*





Selon vous, quels peuvent être les risques de l'IA au niveau de l'automatisation des tests et du métier de testeur ?

Il y a énormément de risques avec l'IA. Le premier est un biais lié à une confiance excessive : si l'IA le dit, l'IA a raison donc je fais ce que l'IA dit. C'est souvent lié au fait qu'on n'a pas le temps. **L'IA devrait être un outil d'aide à la décision, mais l'IA ne devrait pas prendre la décision.** Si on n'a pas le temps de réfléchir, de voir les indicateurs, de voir les données, on n'est plus sûr de l'aide à la décision, on est sûr de la prise de décision. Et c'est ça le vrai risque avec l'IA, sachant que l'IA n'est pas parfaite. Elle vit sur des données, des données qui, même si elles sont mises à jour, pertinentes et représentatives, peuvent changer du jour au lendemain. Prenons un exemple très actuel, les sondages des élections présidentielles : Macron était à 20/25 % dans les sondages, puis il y a une nouvelle donnée, la guerre en Ukraine, et il passe à 30/35, puis il y a une nouvelle donnée, les cabinets de conseils, et il repasse à 25 %. La vérité d'un jour n'est pas la vérité du lendemain et ça l'IA n'est pas capable de l'anticiper.

Personnellement, quel est votre degré de confiance en l'IA ?

Par défaut, je n'ai pas confiance. Je vois du potentiel, des possibilités, des contextes où ça va fonctionner.

Par exemple, pour Gravity®, on a deux retours d'expériences sur les API qui vont être donnés à la JFTL 2022 (Journée Française du Test Logiciel). Dans ce contexte, dans ce cadre-là, ça fonctionne bien et donc, je vais avoir confiance, pour un contexte similaire, de réussir à implémenter une solution qui va faire gagner du temps. Sinon, il n'y a pas de confiance tant qu'il n'y a pas de retour d'expérience, par défaut, ça ne marche pas.

Mais alors, quel serait le contexte idéal pour que ça marche ?

Le contexte idéal ça va être tout ce qui est tâches répétitives, rébarbatives, des choses qui ne sont pas forcément difficiles mais faciles et peu intéressantes à faire. Elles vont requérir de faire beaucoup de choses faciles, mais l'enchaînement va être fatigant et devenir difficile. Tout ça va être sujet à automatisation et c'est le cas dans toute l'histoire de l'humanité d'ailleurs : les outils, on les a faits car c'était plus facile de casser une noix de coco en ayant une pierre aiguisée. Et à chaque fois, on s'est dégagé de plus en plus de temps.





« Nous allons arriver à développer des systèmes de tests intelligents et autonomes. (...) Le sujet n°1 c'est le test de régression, des tests qui sont pénibles à maintenir, à faire. »

Interview Bruno Legeard

Bruno Legeard est un expert de l'automatisation des tests, qui travaille depuis plusieurs années sur l'apport de l'IA pour faciliter les tests logiciels. Professeur de Génie Logiciel à l'Université de Franche-Comté, il encadre plusieurs thèses de Doctorat sur ce thème. Il est aussi l'un des co-fondateurs de Smartesting, qui développe des outils d'automatisation des tests à base d'IA, et il a contribué au sein de l'ISTQB à la nouvelle certification AI Testing.

Vous avez animé une conférence sur l'utilisation des traces d'exécution pour optimiser les tests automatisés de régression avec l'IA, pouvez-vous nous en dire un peu plus ?

À partir du moment où on a pris conscience de ce besoin de l'automatisation des tests, la question c'est : "comment faire" ? Les pratiques courantes qui s'appuient sur le codage des scripts automatisés, sont des pratiques qui posent des difficultés aujourd'hui car elles induisent un effort de maintenance très important. Et c'est là qu'intervient l'apport de l'IA et de nouvelles technologies, pour réduire cet effort de maintenance, d'une part et pour s'assurer que les tests automatisés couvrent les parcours clés en production. C'est ce sujet-là que nous avons abordé et pour lequel nous avons proposé une solution basée sur l'IA qui est l'analyse des traces d'usage du logiciel pour compléter la couverture des tests automatisés. Tester les parcours qui comptent, en sachant le faire à partir de l'analyse de ce que fait vraiment l'utilisateur.

Hormis cet effort de maintenance, quels sont les autres gains que l'IA peut apporter à l'automatisation des tests ?

Il y a un ensemble de techniques qui commencent à apparaître, basées sur l'IA, sur plusieurs sujets liés à l'automatisation des tests. Par exemple, le fait de pouvoir identifier, lorsque le test automatisé est un test classique (un test de bout en bout web par exemple), lorsque ce test simule l'utilisateur, cela nécessite de pouvoir activer des objets de l'interface. Et souvent, les tests cassés viennent d'un changement sur ces objets graphiques. Et l'apprentissage automatique (machine learning) permet de retrouver le bon objet et de modifier automatiquement le script pour réparer le locator, c'est-à-dire la localisation de l'objet graphique qui doit être activé durant le test. Mais ça, ce n'est qu'un élément.

Le deuxième sujet, c'est celui de la pertinence des tests automatisés, c'est-à-dire s'assurer en permanence que les parcours utilisateurs clés sont bien couverts par les tests automatisés. Et il y a un troisième sujet important, qui est le sujet de la priorisation des tests à l'exécution. Lorsque vous avez des **tests de bout en bout***, qui assurent que les parcours utilisateurs fonctionnent, l'exécution peut prendre du temps. Une des solutions à ce problème, c'est de pouvoir prioriser l'exécution des tests automatisés en fonction de l'apprentissage des résultats précédents, d'une prédiction de l'anomalie qui va être détectée.

L'idée, donc, c'est d'exécuter des scénarios* qui seront plus souvent KO pour donner un statut rapide sur, par exemple, une correction d'erreur ?

Absolument. La magie de l'IA et du machine learning, c'est que ce ne sont pas des règles qu'on va humainement définir. C'est l'apprentissage du modèle, des algorithmes par les algorithmes, qui permet de prédire avec une bonne fiabilité. Imaginons qu'on ait 300 tests à exécuter, il y en a 17 qui vont échouer et révéler des anomalies. Avec la prédiction, ces 17 tests vont alors se retrouver dans les 20 premiers à exécuter. Si on a le temps d'exécuter ces 300 tests en 3 minutes, on ne va pas en avoir besoin, mais la réalité n'est souvent pas celle-là. La réalité, c'est que les 300 tests s'exécutent en 4/5h et on a donc un gros intérêt à avoir une priorisation fiable. Et l'IA, dans nos travaux, c'est une technique d'apprentissage supervisé, le fait qu'on apprenne à partir de l'historique, du contexte dans lequel le test s'exécute, de sa durée, des changements qui ont été fait, etc. Il faut voir l'IA comme un facilitateur au quotidien pour les testeurs.

**Test de bout en bout : Tests permettant de tester les interactions et fonctionnalités d'un parcours au plus proche de ce que réaliserait un utilisateur final.*

**Scénario : Déroulé des différentes étapes à réaliser pour tester une ou plusieurs fonctionnalités sur une application web ou mobile.*



Au contraire, pour vous, quelles sont les limites de l'IA dans l'automatisation des tests ?

Dans quelques années, ça me paraît assez clair que nous allons arriver à développer des systèmes de tests intelligents et autonomes. Il y a des choses en laboratoire qui commencent à apparaître autour de tests auto-adaptatifs. Le sujet n°1 c'est le test de régression, des tests qui sont pénibles à maintenir, à faire. Je parierais que d'ici 5 à 10 ans, on aura ce type de robot. On aura des capacités qui remplaceront la création des tests par des testeurs dans certains environnements et de façon automatique. Cela veut aussi dire les profils, les bonnes connaissances métiers, etc, c'est du travail un peu répétitif qui sera remplacé pour partie. Mais, les testeurs ont une très forte valeur ajoutée sur l'ensemble des activités d'ingénierie de la qualité, et la mise en place des robots de tests intelligents et autonomes viendra comme une aide et un renforcement de leurs activités.

L'IA est un sujet assez complexe, selon vous, quels sont les défis que cela peut poser dans l'automatisation des tests ?

Lorsqu'on utilise et qu'on met en œuvre des techniques d'apprentissage à partir de sources de données, si les sources de données ne sont pas fiables, alors les résultats de l'IA ne seront pas fiables. Si on cherche à prédire des taux d'anomalies ou des risques associés à différentes composantes logiciel en fonction de l'historique des développements, encore faut-il que ces données d'historique aient été archivées, maintenues et qu'elles soient fiables. C'est aussi une problématique, il ne suffira pas de dire : je n'ai qu'à prendre ce moteur d'IA pour avoir tel résultat, comme par exemple, faire une prédiction sur le taux d'erreur du composant. Si on n'a pas les bonnes données en amont, le composant ne donnera jamais rien. Les testeurs ont d'ailleurs un rôle à jouer dans le fait que les données utilisées par l'IA soient de bonne qualité.





Interview Xavier Blanc

« Faire une IA qui va répondre à tous les problèmes de la Terre, ce n'est même pas une ambition. Ce qu'on va plutôt chercher à faire, c'est une IA capable de s'adapter au contexte. »

Xavier Blanc est un spécialiste en qualité logicielle. Directeur du Laboratoire Bordelais de Recherche en Informatique (LaBRI), il a co-fondé ProMyze, une start-up qui permet de définir et échanger les bonnes pratiques de développement logiciel et d'améliorer ainsi la qualité des équipes. Il est également Professeur à l'Université de Bordeaux, avec une spécialisation en génie logiciel.

L'IA est un sujet au cœur des tendances actuelles et il commence notamment à s'inviter dans le monde du test. Personnellement, comment voyez-vous l'IA s'inscrire dans l'automatisation des tests ?

L'IA peut notamment aider l'automatisation en ne laissant pas l'automaticien tout seul : comment assister l'automaticien dans son travail pour l'aider à fournir des tests de meilleure qualité ? Parce que finalement, quand on écrit un test automatisé, on répète des opérations et c'est là où l'IA a montré qu'elle était très forte. Quand on commence à répéter, quand il y a des données qui existent et qu'on fait des choses qu'on a déjà faites, ou ont été faites par d'autres personnes, c'est là que l'IA peut nous aider à ajouter de la qualité. On va prendre une application et on va demander à l'IA de l'analyser et nous aider à trouver les tests les plus pertinents.

Il y a un deuxième point sur le diagnostic. On a notre base de tests, on sait l'automatiser, ça s'exécute très bien, et puis parfois il y a encore un diagnostic qui est difficile à faire. Le test nous dit que ça ne passe pas, c'est rouge donc il y a un bug, mais ce n'est pas un bug, c'est quelque chose qui est contextuel : c'est peut-être le réseau qui ne marche pas bien, une latence. Et on perd du temps sur le diagnostic. On a de l'expérience mais c'est tellement compliqué que l'intuition ne suffit pas et c'est là où l'IA peut rapidement nous apporter de gros bénéfices.

Quels sont les challenges que peut poser l'IA au niveau de l'automatisation des tests ?

L'IA va avoir du mal à être très contextuelle. Les applications qu'on teste se ressemblent, mais elles ont toutes des spécificités.

Ces spécificités font que les choix qui s'appliquent à certaines applications ne s'appliquent pas à d'autres applications. Faire une IA qui va répondre à tous les problèmes de la Terre, ce n'est même pas une ambition. Ce qu'on va plutôt chercher à faire, c'est une IA capable de s'adapter au contexte. Il va donc falloir qu'elle apprenne très rapidement, qu'elle devine les spécificités de ce contexte et qu'elle soit très pertinente. Et c'est là où on va avoir d'énormes difficultés, dans les mois ou les années à venir, c'est de trouver ce contexte et savoir prendre les bonnes décisions. Actuellement l'IA est forte quand elle a plein de points : elle sait trouver où la courbe qui passe entre tous ces points. Le challenge va être d'enlever les points qui n'ont pas de sens dans le contexte d'utilisation pour telle ou telle application.

Quel serait le contexte idéal pour que l'IA s'inscrive dans l'automatisation des tests, dans ce cas ?

J'aimerais que l'IA soit embarquée dans les applications. On a des masses de données mais on passe du temps à les analyser. Cela crée un décalage. On fait tourner les algorithmes et on arrive à avoir une IA performante, mais ça manque un peu de réactivité. Le fait d'avoir l'IA directement dans l'application permettra qu'elle soit très réactive, qu'elle apprenne directement à partir du comportement des utilisateurs. Les comportements des utilisateurs changent du jour au lendemain, c'est quasiment impossible de les prédire et on a toujours un petit train de retard. Je pense qu'on va avoir des IA qui seront embarquées dans les applications et qui vont minimiser le temps de cette boucle de rétroaction. L'IA va pouvoir donner des préconisations : les utilisateurs font ça aujourd'hui, vous devriez regarder ce genre de tests, etc. On le voit déjà dans la sécurité. Ils sont tout le temps en train de monitorer et cela leur permet de déceler de nouvelles attaques, en live. Ils ont aussi de l'IA collaborative : ils ont plusieurs sites et ils essaient de mettre en commun les connaissances. On pourrait imaginer ce genre de choses entre différentes applications. Cela permettrait de se rendre compte de changements dans les comportements utilisateurs.

Selon vous, en quoi l'IA pourrait aider les équipes métier, qui n'ont pas de profil technique, dans l'automatisation des tests ?

Si on a une IA dans l'application, il faudrait qu'elle soit capable de parler deux langues : la langue des gens du métier et la langue technique des personnes qui travaillent au développement logiciel. Si on a suffisamment d'intelligence dans l'IA, on pourrait imaginer qu'elle comprend mieux les utilisateurs. Elle pourrait ainsi aller vers les gens du métier en leur disant ce que font exactement les utilisateurs. Pour un site web, cela permettrait de savoir ce que font réellement les utilisateurs et de réagir en temps réel.

Cela permettrait aussi aux gens du métier de faire des tests de simulation : que se passerait-il si on mettait ces nouvelles fonctionnalités en place ? Actuellement, on pose des métriques et on les regarde, mais on pourrait aussi simuler pour savoir si ça vaut le coup ou pas. Et si on met en place des nouvelles fonctionnalités, quelle serait la prédiction de l'IA ? Est-ce que ça va être intéressant ou pas ? On pourrait aussi imaginer la même chose côté technique : et si j'utilise ce nouveau framework, est-ce que je vais introduire des nouveaux bugs ou pas ? Quels seraient les tests dont je vais avoir besoin ?

Quels risques peut poser l'IA au niveau du monde du test ?

L'IA est très forte pour trouver la courbe qui passe par des points. Si on lui demande de faire une prédiction (quel est le prochain point ?), elle va nous donner le prochain point sur la courbe qu'elle a tracée. Le problème, c'est que, si on ne prend pas les bons points, l'IA va être très mauvaise.

Un problème avec l'IA, c'est qu'il y a peut-être des points qui ne sont pas sur la courbe qu'elle a tracée et qui pourtant sont intéressants. Il va falloir peut-être débrancher l'IA et prendre des risques. Les gagnants seront-ils ceux qui auront réussi à prendre des risques à contre-courant des prédictions de l'IA ?

Il y a donc deux écueils : avoir une IA qui n'est pas très maline, qui a appris sur des mauvais points, et qui donc nous envoie dans la mauvaise direction, ou avoir une IA qui est maline mais qui manque d'originalité, et qui ne va pas nous proposer de prendre un point qui n'est pas dans la bonne direction et qui, pourtant va être un succès dingue. L'IA fait des moyennes avec le risque de faire des prédictions très mainstream.

Justement, quel est votre degré de confiance envers l'IA ?

J'ai absolument confiance en l'IA, si on considère que l'IA c'est l'algorithme qui trouve la courbe qui passe par tous les points. Ce n'est pas une réponse très honnête, mais je n'ai pas envie de taxer l'IA parce qu'on lui a donné des mauvais points. L'IA apprend donc si je lui donne des choses pas intelligentes, elle va apprendre des bêtises. La difficulté, c'est de lui donner des choses intelligentes sur lesquelles elle peut apprendre. On a aussi des algorithmes qui commencent à comprendre dans quel sens il faut donner les informations pour qu'elle apprenne vite, mais de là à demander à l'IA de faire le tri elle-même sur les informations qu'elle doit utiliser et qu'elle doit rejeter, je pense que ce n'est pas à l'algorithme de l'IA de faire ce tri, c'est en amont, c'est là où il va falloir qu'on soit un peu plus exigeant sur notre façon de remonter des données. Donc oui, je suis confiant, et je pense qu'on arrivera à des taux de confiance encore plus importants si on est capable de donner les bons points. Est-ce que j'ai confiance aux points qu'on donne, beaucoup moins, mais pour moi, ce n'est plus de la responsabilité de l'IA.

Est-ce que vous voulez nous parler un petit peu des travaux que vous faites sur le test et l'IA ?

On travaille beaucoup sur ce que font les utilisateurs des applications web. On commence à avoir pas mal de sondes qui permettent de mesurer ce qu'ils font. On a des modèles qui nous expliquent qui sont nos utilisateurs. Il y a plein de technologies qui sont connues pour faire des sortes de classifications : il y a l'utilisateur mainstream, l'utilisateur qui est très rare, etc. Avec nos modèles, l'idée c'est de fournir des jeux de tests automatisés selon les utilisateurs : si tu veux tester ce que fait l'utilisateur lambda, il suffit que tu exécutes ça, si tu veux tester les utilisateurs un peu tendance, voilà plutôt ce qu'il faudrait que tu fasses. Le travail, ici, qui est un peu difficile, c'est de prendre tous les points que font les êtres humains sur les applications et essayer de voir si on ne peut pas classifier ça, si on ne peut pas catégoriser ces points et demander à l'IA de fournir différentes courbes. Une fois qu'on a ces modèles de comportements, on va pouvoir se poser la question : est-ce que ces modèles ne vont pas nous aider à mieux tester l'application ? Cela rejoint tout ce que j'ai dit, c'est qu'actuellement, ces modèles de comportement sont faits à posteriori et on aimerait bien les avoir en temps réel, pour savoir s'ils ne changent pas tous les jours, en intégrant l'IA dans l'application directement.

CONCLUSION

Nous l'avons vu ensemble à travers ce guide : de nos jours, et grâce à l'automatisation codeless, l'automatisation des tests peut être mise en place simplement et par tous, sans compétences techniques. Mais faut-il encore bien la mettre en place.

L'automatisation des tests peut être un véritable facilitateur pour vos équipes au quotidien, à condition d'élaborer en amont une stratégie, et surtout la bonne, celle qui est adaptée à votre entreprise.

Pour réussir, il est important de suivre toutes les étapes clés que nous vous avons présentées. Commencez par prendre le temps de fixer vos objectifs, en accord avec la vision de votre entreprise. En fonction de vos ressources (financières, humaines, matérielles), il peut être intéressant de vous entourer d'experts ou de partenaires pour vous accompagner. Et surtout, ne faites pas l'impasse sur l'analyse et la maintenance de vos tests automatisés et pensez à suivre vos résultats régulièrement grâce à des KPIs pertinents.

Enfin, comprenez bien que l'automatisation des tests n'est pas uniquement l'affaire des équipes techniques. L'automatisation des tests est utile à tous pour :

- *Réduire votre time to market*
- *Avoir la visibilité sur tous vos bugs*
- *Améliorer votre taux de conversion*
- *Préserver votre image de marque*
- *Décharger vos équipes*
- *Gagner du temps*

Alors, prêt à vous lancer dans une démarche d'automatisation des tests ?

Et maintenant ?

Devinez qui pourrait vous accompagner dans l'automatisation de vos tests ?

MR SURICATE !

Nous détectons les bugs sur les parcours client des sites web et mobiles. Nous avons pour cela conçu une solution SaaS codeless facile et rapide à mettre en place, adaptée à un profil non-technique, accessible 100 % en ligne et sans intégration. Nous travaillons avec de belles enseignes nationales telles que Fnac Darty, KFC, Zadig et Voltaire, etc.

L'ambition de Mr Suricate est d'ouvrir l'automatisation des tests à tous et notamment de permettre au métier de prendre la main sur un secteur trop accoutumé à l'IT.

Ne vous prenez plus la tête avec vos bugs, optez pour Mr Suricate et l'automatisation des tests devient un jeu d'enfants !

Retrouvez-nous !



Mr Suricate

7 rue Mathurin Brissonneau 44100 Nantes - suricate@mrsuricate.com - www.mrsuricate.com

