



AGILITEST

EBOOK

AUTOMATISER
LES TESTS
FONCTIONNELS
POUR DES
ÉQUIPES
HEUREUSES

WWW.AGILITEST.COM

SOMMAIRE

A propos de l'auteur	3
1. Des tests manuels aux tests automatisés	5
a. L'organisations des équipes de test manuel : le cycle en V	6
b. Les challenges rencontrés par les tests manuels	8
c. Comment réorganiser les équipes pour permettre l'automatisation des tests	10
2. Pour une approche plus intelligente des tests automatisés	12
a. Identifier les tests à automatiser	13
b. Comment implémenter le test continu	16
c. Les gains à tester fréquemment	18
3. Trouver les bons outils	20
a. Comment trouver le bon outil d'automatisation des tests	21
b. Pour aller plus loin : construire sa stratégie d'automatisation des tests	25
A propos d'Agilitest	26



AGILITEST

A PROPOS DE L'AUTEUR

L'AUTEUR

Marc Hage Chahine

Marc Hage Chahine, expert test méthodes et outils chez Sogeti, est dans le monde du test et de l'Agile depuis qu'il a commencé à travailler en 2011.

Il partage sa passion, sa vision et ses connaissances à travers ses écrits dont la majorité sont sur le [blog de la taverne du testeur](#).

Au-delà des articles, il participe à l'organisation d'événements liés au tests comme la JFTL (Journée Française des Tests Logiciels) et la STLS (Soirée du Test Logiciel à Sophia).



AGILITEST



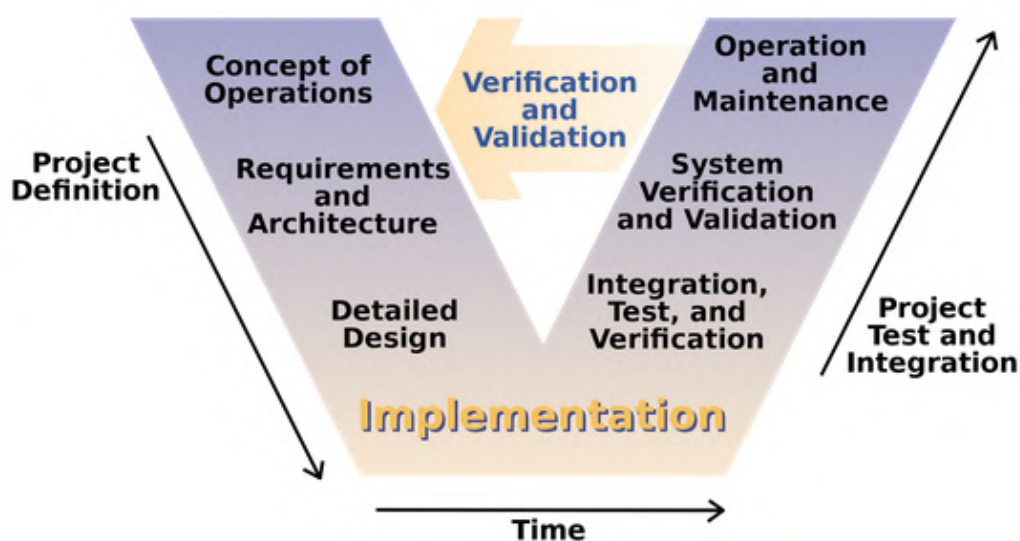
AGILITEST



DES TESTS MANUELS AUX TESTS AUTOMATISÉS

a. L'organisation de l'équipe chargée des tests manuels : le cycle en V

Pendant de nombreuses années, les logiciels ont principalement été développés avec une méthodologie de Cycle en V. Pour rappel, le cycle en V est un modèle d'organisation d'un projet qui se caractérise par le flux d'activité qui détaille le produit jusqu'à sa réalisation et un flux qui assemble le produit en vérifiant sa qualité.



The V-model of the Systems Engineering Process (en.wikipedia.org)

Dans ce modèle, plus les étapes interviennent en amont, plus la validation de l'étape se fait tardivement. Ce constat a été établi par des experts souhaitant mettre en place le "Shift Left", qui sert à détecter les anomalies le plus tôt possible, en implémentant des processus comme les revues. Pour rappel, les revues sont des processus de vérification de livrables écrits qui peuvent s'adapter à de très nombreux sujets comme le code, les spécifications, les tests scriptés...

En plus des tests réalisés tardivement, le Cycle en V pose d'autres problèmes. Les cycles de livraisons sont longs et limitent ainsi le nombre de livraisons par an (souvent entre une et quatre). Les "phases" de test spécifiques se déroulent aussi de manière très ponctuelle. Un autre problème se pose avec les SUT (System Under Test) qui ont de nombreux changements à tester, avec de nombreuses exigences, induisant des campagnes de tests qui peuvent durer plusieurs semaines.

Cette méthode de cycle en V donne l'avantage aux tests manuels. En effet, les tests de régression sont peu exécutés, une seule fois par version, ce qui rend leur automatisation difficilement rentable. Les équipes de tests qui travaillent en cycle en V passent beaucoup de temps à préparer une campagne avec des phases de conception et d'écriture des tests. Suite à cela, la campagne de test peut commencer, après la livraison du logiciel. Cette campagne de test pouvant être longue, un suivi est souvent mis en place pour avoir des bilans intermédiaires. À la fin de la campagne, l'équipe de test passe à un autre sujet, comme la préparation d'une nouvelle campagne pour un SUT différent.

Les charges de test sur un SUT en particulier ne sont donc pas du tout lissées avec la méthode du Cycle en V. Cependant, une bonne compréhension du métier, la capacité à comprendre les spécifications et à concevoir les tests sont une nécessité. En cycle en V, il y a beaucoup moins de droit à l'erreur comparé à la méthode Agile, où une nouvelle version peut être livrée très rapidement.

Cette méthodologie a donc formé des testeurs très compétents dans l'art de comprendre le métier, mettre en place des stratégies autour du RBT (Risk Based Testing) et concevoir des tests de qualité.

Les testeurs habitués au cycle en V n'ont généralement pas été amenés à travailler sur l'automatisation des tests. En effet, il y a peu de campagnes exécutées pour qu'elles soient jugées répétitives et nécessitant de l'automatisation.

b. Les défis rencontrés par les tests manuels avec la méthode Agile

La méthode Agile est totalement différente des méthodologies comme celle du cycle en V. En effet, en Agile, on ne parle plus “projet” mais de “produit”.

En Agile, le logiciel est construit de manière itérative ; ce qui a de nombreuses conséquences qui l’opposent au cycle en V :

- Les cycles de livraisons sont courts et obligent de multiples livraisons chaque année (souvent 10 au minimum).
- L’absence de “phase” de test persiste, mais du test en continu est fait par des équipes pluridisciplinaires.
- Les SUT sont peu modifiés ; les phases de test doivent donc être courtes... tout en incluant la régression.

Le défi de la méthode Agile est la multiplication des campagnes qui doivent rester courtes tout au long de la vie du produit. Du point de vue de la validation des fonctionnalités, cette contrainte n’est pas réellement problématique puisque l’Agile propose des petits changements ; notamment avec le principe INVEST pour les User Story (US). Ce n’est cependant pas le cas pour le produit dans son ensemble.

Un produit développé en Agile peut être très complexe et avec de nombreuses fonctionnalités. La campagne de régression liée à ce produit peut être très importante en termes de volume ou de nombre de tests. Cette campagne a tendance à “grossir” à chaque nouvelle livraison, rendant la campagne toujours plus longue à exécuter ; surtout si cela est fait de manière manuelle.

Au-delà de l'aspect rébarbatif des campagnes de régression qui, à elles seules, poussent à automatiser, gérer sa régression avec des tests uniquement manuels en Agile est quasiment systématiquement source d'échec pour le développement d'un produit, à moyen ou long terme.

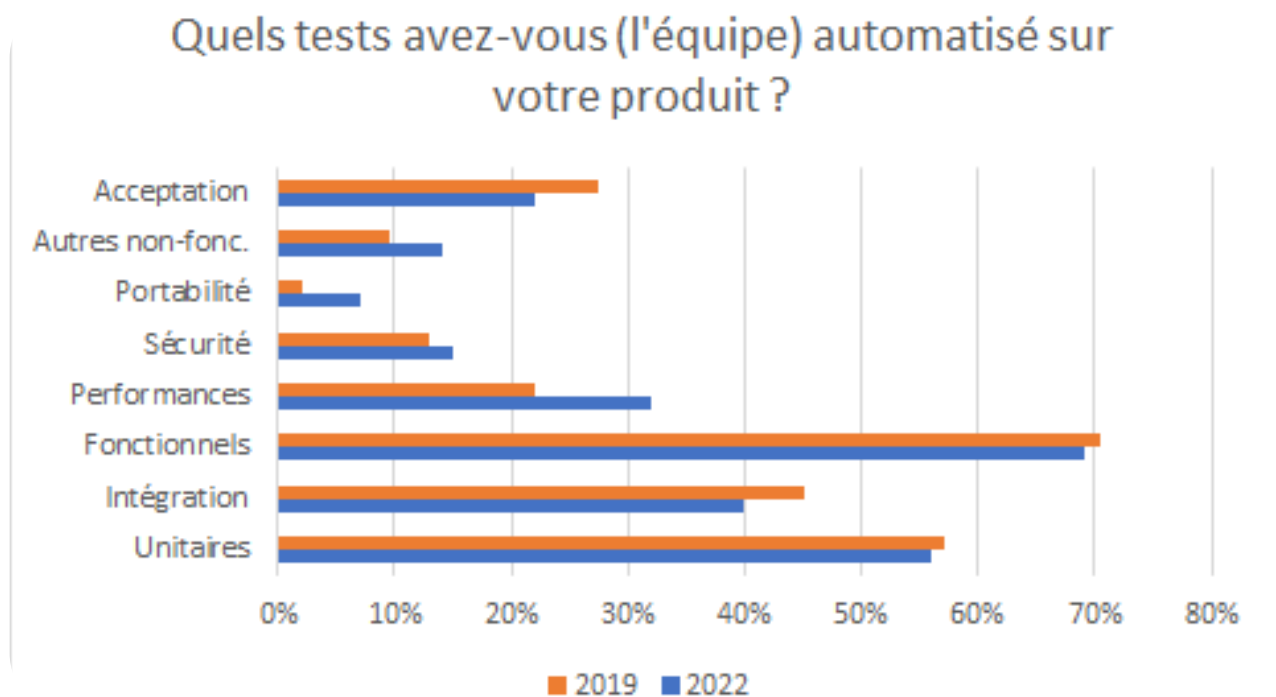
Cet échec peut prendre alors deux formes différentes :

- La première et la plus fréquemment rencontrée : une diminution de la qualité en production et des utilisateurs mécontents, car trop peu de tests de régression sont exécutés.
- La seconde est plus rare : un refus de faire une croix sur la qualité du produit. La campagne de régression est de bonne qualité, les tests sont maintenus et exécutés. Malheureusement, cette régression coûte très cher en temps et impacte fortement la capacité de l'équipe à livrer de nouvelles fonctionnalités. Les délais de livraisons sont alors comparables à ceux du cycle en V, mais avec de plus petites livraisons.

Afin d'éviter ces problématiques, il est essentiel d'automatiser autant que possible les tests de régression. L'automatisation est un investissement, car cela demande la mise en place d'un outil et quasi systématiquement, en dehors des tests techniques, davantage de temps à l'écriture que des tests manuels. Néanmoins, ils font gagner énormément de temps lors de l'exécution des tests, ce qui, en Agile, est la problématique principale contrairement au cycle en V.

c. Comment réorganiser les équipes pour permettre l'automatisation des tests

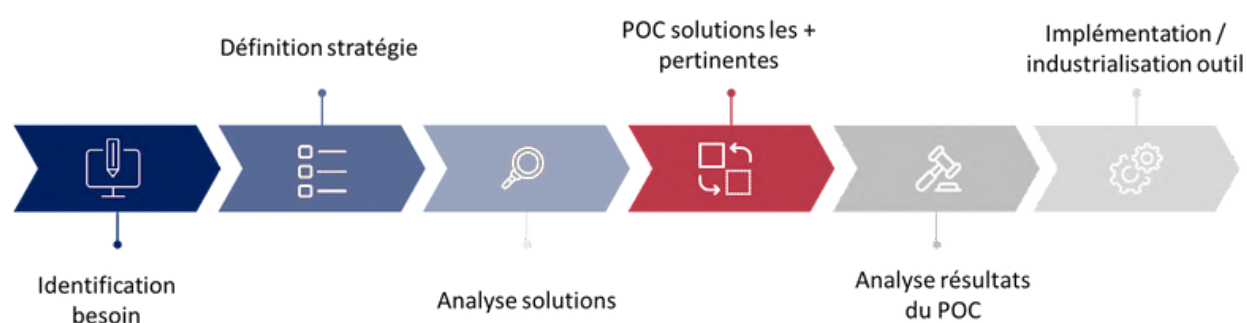
Même si l'automatisation des tests est identifiée comme étant essentielle en Agile, elle n'est malheureusement pas toujours (bien) implémentée, et peut avoir des difficultés à s'imposer. Un sondage de La taverne du testeur « Quel testeur Agile êtes-vous ? » l'illustre bien :



Cette difficulté prend racine dans l'historique des méthodologies de développement des logiciels. Les profils les mieux placés pour tester un logiciel sont les testeurs. Mais ces testeurs n'ont pas forcément développé les compétences nécessaires à l'automatisation des tests. En effet, ces compétences se rapprochent en général des compétences liées au développement.

Pour remédier à ce problème, il y a aujourd'hui de plus en plus d'automaticiens de tests. Ces derniers sont cependant toujours trop peu nombreux pour être présents dans les équipes avec des besoins d'automatisation des régressions. Beaucoup de ces automaticiens viennent du monde du développement ce qui peut entraîner, dans certains cas, un besoin d'approfondissement des compétences de testeurs ou au moins des moments d'échange avec l'équipe autour des tests à automatiser.

Il est ainsi nécessaire que les équipes se réorganisent afin de pouvoir automatiser leurs tests de régression. Il existe de nombreuses méthodes, chacune plus ou moins adaptée au contexte. Ces méthodes ont cependant un point commun : la nécessité de ne pas foncer tête baissée dans l'automatisation des tests et de prendre le temps de bien la préparer en faisant, par exemple, une stratégie d'automatisation et des POC (Proof Of Concept) sur des outils. Cette stratégie peut ressembler à cela :



Source : Tutoriel JFTL 2022 : Choisir l'automatisation « pour tous »

La grande différence entre les méthodologies est le choix fait sur l'outil et l'identification des personnes en charge de l'automatisation. Cela peut par exemple être :

- **Un testeur fonctionnel** : dans ce cas, il est important d'avoir un outil peu technique et facile à prendre en main. On peut penser à un outil no-code comme [Agilitest](#).
- **Un testeur et un développeur qui travaillent conjointement** : le testeur concevant les tests, le développeur les automatisant avec le testeur à ses côtés.
- **Un automaticien de test** ayant l'ensemble des compétences requises et pouvant se concentrer sur le choix d'un outil répondant au besoin en termes de capacité à automatiser les technologies, mais aussi en termes de productivité.

De manière générale, il est quand même important que l'automatisation ne soit pas intégralement portée par une unique personne de l'équipe car son absence se révèle alors très vite préjudiciable. Il faut alors bien identifier un ou plusieurs back-ups pouvant prendre son relais en cas de besoin. Ce besoin de back-up donne alors plus d'importance au fait d'avoir un outil d'automatisation de test accessible.



**POUR UNE
APPROCHE PLUS
INTELLIGENTE DE
L'AUTOMATISATION
DES TESTS**

a. Identifier les tests à automatiser

L'automatisation de l'exécution des tests, généralement simplement appelée automatisation des tests, est un outil dont le rôle est de gagner du temps sur l'exécution des tests.

Dans le cas des tests d'interface graphique, qui sont les tests auxquels les personnes pensent le plus quand on parle d'automatisation, mais qui sont également les plus compliqués à mettre en place, il est particulièrement important de bien sélectionner les tests que l'on souhaite automatiser.

À noter : l'automatisation des autres types de test comme les tests d'API suivent à peu près les mêmes critères, mais leur automatisation est souvent plus aisée et leur retour sur investissement beaucoup plus rapide.

- **Automatiser l'exécution de tests réguliers**

Pour bien choisir les tests à automatiser, il faut avant tout sélectionner des tests qui ont pour vocation d'être exécutés plusieurs fois. Il n'y a généralement que peu d'intérêt à investir du temps dans l'automatisation d'un test qui ne sera exécuté qu'une seule fois.

Il faut également s'assurer que les tests sélectionnés ne soient pas trop complexes à automatiser avec le contexte : des technologies non couvertes par l'outil d'automatisation, la nécessité d'une intervention extérieure, l'accès à des comptes ou messageries sécurisées, des problématiques d'environnement ou de données... Ce sont des problématiques pouvant engendrer des difficultés d'automatisation. Il est important d'avoir l'objectif d'automatiser ses tests réguliers, et, dans le cas d'une non-automatisation, de pouvoir expliquer la raison de cette dernière. Cet état d'esprit est essentiel. En effet, ne pas automatiser par défaut amène souvent à un fort ralentissement dans la capacité de livraison ou à une baisse de la qualité avec des tests manuels moins souvent exécutés.

Il est également important de rappeler que l'absence de l'automatisation de certains tests de la campagne de régression est un frein majeur à la mise en place du déploiement continu et du test continu en général.

Malheureusement, l'automatisation de tests réguliers n'est pas une garantie suffisante de la réussite de l'automatisation. La réussite de l'automatisation de ses tests dépend aussi de la qualité et de la pertinence des tests automatisés.

- **Construire une campagne de tests automatisés pertinente**

L'exhaustivité des tests est impossible. L'automatisation permet d'en exécuter un nombre conséquent dans un temps limité, mais il peut vite s'avérer inutile d'ajouter certains tests si ces derniers n'ont pas ou peu de valeur ajoutée.

En effet, une des raisons de l'échec de l'automatisation des tests peut être les tests en eux-mêmes. Les tests automatisés d'IHM sont très majoritairement des tests appartenant à des campagnes de régression, dont le but est de détecter si l'ajout de nouvelles fonctionnalités a engendré des régressions majeures sur notre logiciel. Ce filet de sécurité que représente cette campagne de régression se doit donc de couvrir les bons scénarios, de faire les bonnes vérifications, mais aussi d'évoluer pour lutter contre le paradoxe du pesticide.

En ce qui concerne les bons scénarios, le réflexe est de se concentrer sur les usages. Une régression apparaissant sur un parcours jamais utilisé ne sera vraisemblablement pas une régression majeure. En revanche, un usage important non couvert par les tests et victime d'une régression peut vite s'avérer être la source d'un grave incident en production. Afin d'éviter ce problème, il est très important de bien connaître ses utilisateurs et leur manière de manipuler le logiciel.

Une bonne couverture des usages peut permettre de “remplir sa campagne de régression” à 80 %, voire à 90 %. Les pourcentages restants étant pour des tests plus spécifiques comme des tests de sécurité, des tests liés à des parcours peu empruntés mais primordiaux (ex : les interface d'administration) ou des tests réglementaires (ex : RGAA pour l'accessibilité). Cette sélection des tests par l'usage est un moyen efficace de lutter contre l'illusion d'absence d'erreur, mais aussi de bien gérer son effort de test.

Pour les bonnes vérifications, il est important de bien identifier le but du test et de s'assurer que cette vérification ne souffre d'aucune ambiguïté. Ce qui fonctionne de manière manuelle ne fonctionne pas forcément avec l'automatisation. De même, il est important lors de l'analyse des tests en échec de trouver facilement d'où vient l'anomalie, mais aussi d'être sûr que cet échec soit “pertinent” et que nous n'ayons pas affaire à un flaky test (faux positif). Pour éviter cela, il y a de nombreuses bonnes pratiques possibles. Une pratique intéressante peut être de vouloir écrire plus de tests, avec chacun un objectif bien défini, plutôt qu'un seul test avec de nombreuses vérifications. C'est souvent le cas avec les tests manuels.

Enfin, une campagne de régression se doit d'évoluer. Elle évolue évidemment avec de nouveaux tests qui viennent l'enrichir à mesure que le produit évolue, mais ses tests plus anciens doivent également évoluer. Par le mot “évolution”, il ne faut pas entendre “maintenu” ni “mis à jour”. La maintenance est une obligation dans toute campagne de TNR, sous peine d'avoir des tests en échec sans raison valable. Par “évolution”, il faut comprendre que les tests ne découvrant pas d'anomalies doivent être modifiés afin de parcourir un chemin différent et potentiellement détecter de nouvelles anomalies que le test initial n'aurait jamais trouvé. Cette problématique est d'ailleurs tellement connue qu'elle fait l'objet d'un des 7 principes du test à part entière : le paradoxe du pesticide. À noter que même une campagne créée sur la base des usages peut être sujette à cette problématique. Elle y est même encore plus sensible, car les usages peuvent évoluer très vite.

b. Comment mettre en œuvre des tests continus

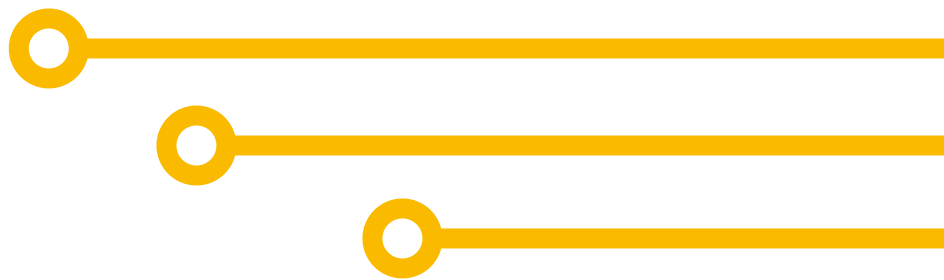
L'automatisation de l'exécution des tests, c'est bien. Mais automatiser le lancement de sa campagne de tests automatisés sur demande et à chaque fois que l'on souhaite valider une modification du code, c'est mieux !

Cette automatisation de l'exécution est souvent décrite sous l'appellation de test continu. Cette terminologie peut être perçue de deux manières différentes. La première est que le test se fait quasiment tout le temps et donc en continu. La seconde est que les tests sont intégrés aux chaînes d'intégration continue, et donc que le terme continu indique cette automatisation. Dans tous les cas, le test continu est une belle mise en œuvre du shift-left et permet de détecter des anomalies au plus tôt.

Néanmoins, il n'est pas toujours simple de mettre en place ce test en continu, et ce, pour plusieurs raisons :

- **Une campagne de tests automatisés** peut durer plusieurs dizaines de minutes, voire quelques heures. Dans ce cas, il est impossible d'intégrer l'ensemble de ses tests automatisés à la chaîne et un travail de sélection des tests automatisés est alors nécessaire.
- **Il peut y avoir de nombreuses problématiques** liées aux environnements. Les tests d'IHM automatisés ont besoin d'un logiciel fonctionnel pour s'exécuter, au contraire d'autres types de tests comme ceux sur la qualité du code ou certains tests unitaires. Leur exécution requiert alors la disposition d'un environnement avec la version du logiciel à tester. Cela se traduit généralement par la création d'un environnement spécifique et de ses données à la volée.

- **Une complexité dans l'ajout à la chaîne d'intégration** pour des raisons techniques ou liées à l'outil. Ce problème se surmonte généralement assez rapidement dès lors où des personnes avec les compétences requises sont identifiées.
- **Une analyse complexe des tests** ou pire, des flaky tests venant bloquer trop régulièrement les demandes de merge. Les flaky tests sont un vrai problème du test continu, car même avec des tests fiables à 99 %, une campagne contenant 100 tests n'a que 36 % de chances de ne pas contenir de test en échec sans réel bug. Environ $\frac{2}{3}$ des merges sont valides, car sans bugs détectés par les tests, ils ne sont pas acceptés. Un workaround fréquent, lorsque l'on est conscient de cette problématique, est de faire en sorte que les tests en échec ne soient plus bloquants... mais cela engendre de gros risques sur la qualité.



c. Les gains à tester fréquemment

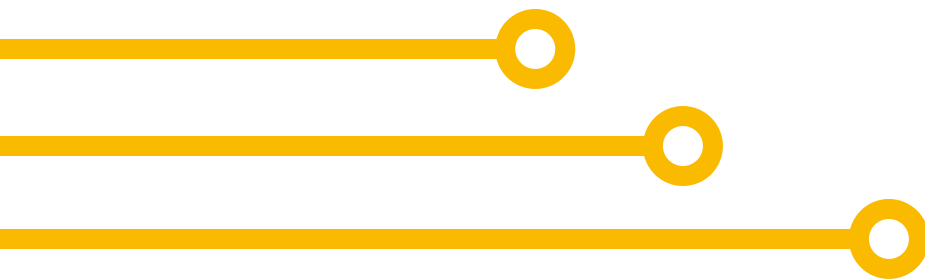
Tester fréquemment manuellement est rarement apprécié par les testeurs. Tester régulièrement de manière automatique permet de nombreux avantages concrets comme ceux du shift-left et sont liés à une détection précoce des anomalies. Tout cela engendre :

- **Un gain de temps pour les équipes** avec une diminution du temps passer à analyser les causes de l'anomalie.
- **Un gain de temps** car le temps nécessaire pour corriger l'anomalie est réduit et une faible probabilité que cette correction engendre des régressions.
- **Un gain de productivité** avec une diminution du "re-travail" ou du travail sur les corrections d'anomalie, ce qui engendre plus de temps pour développer de nouvelles fonctionnalités sur son produit.
- **Un gain de qualité** avec un plus fort % d'anomalies corrigées (car le temps ne manque plus).
- **Un gain de disponibilité** pour des tâches à plus forte valeur ajoutée.

Au-delà de ce qu'apporte le "shift-left", l'automatisation de l'exécution des tests permet aussi de diminuer le stress des équipes lors des déploiements ou lors de la phase d'exécution de la campagne de régression. Le stress est réduit, car les délais d'exécution de la campagne sont bien moindres et les testeurs sont assurés que tous les tests jugés nécessaires ont été exécutés.

De même, ne pas devoir réexécuter sans cesse les mêmes tests est un véritable soulagement pour un testeur. Celui-ci peut en effet très mal vivre sa mission s'il est amené à toujours exécuter manuellement les mêmes tests. Ce soulagement du testeur n'est pas le seul effet positif! En effet, bien implémenter l'automatisation de l'exécution des tests, c'est également l'assurance que tous les tests sont "bien" exécutés avec toutes les vérifications requises. En effet, plus les tests sont répétitifs, plus il est probable d'introduire une erreur humaine suite à un déficit d'attention.

En somme, une automatisation des tests bien implémentée, c'est une équipe passant moins de temps sur des correctifs, une équipe plus productive et plus confiante dans ses livraisons ainsi qu'un testeur plus épanoui et travaillant sur des tâches à plus grande valeur ajoutée. Ainsi, une équipe ayant bien mis en place l'automatisation est une équipe plus sereine et pouvant s'épanouir plus facilement. On remarque dans ce cas que les gains attendus sont, de ce point de vue, assez proches des gains de la pratique du BDD et de leur mise en place à travers les 3 amigos.

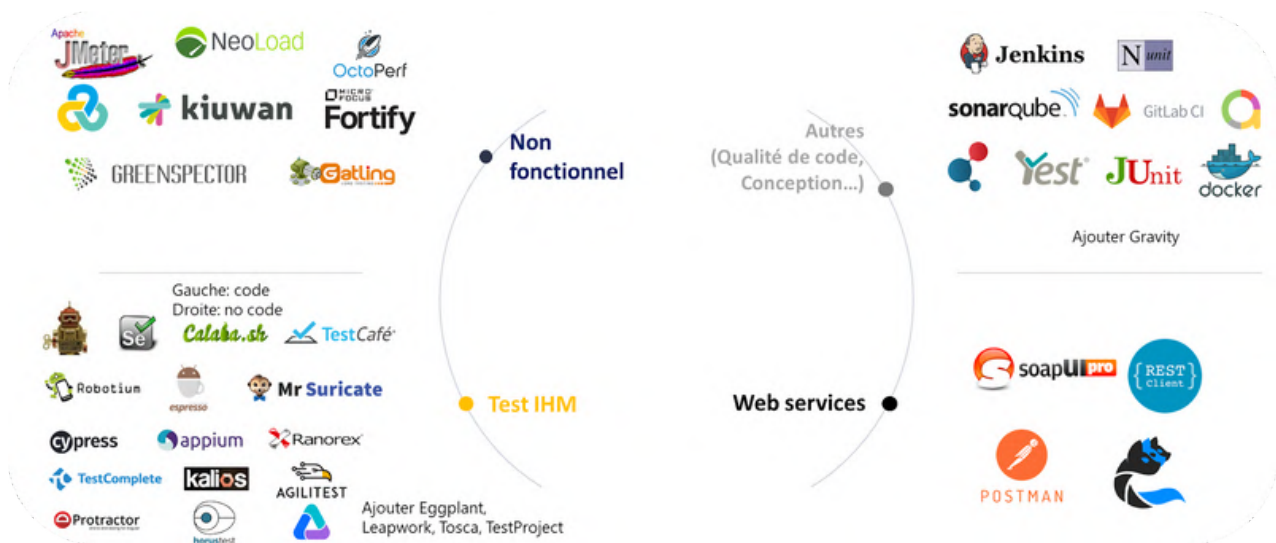




**TROUVER LES
BONS OUTILS**

a. Comment trouver le bon outil d'automatisation des tests

Les outils d'automatisation des tests sont très nombreux sur le marché. Il y en a certainement un voire plusieurs qui répondent à votre besoin et peuvent s'adapter à votre contexte. Voici un échantillon d'outils d'automatisation de test :



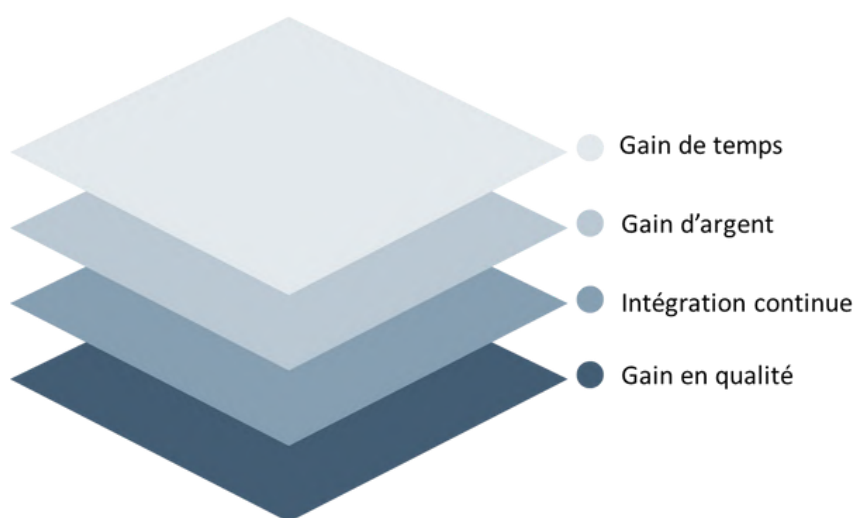
Source: Tutoriel JFTL 2022 : Choisir l'automatisation « pour tous »

Cette diversité, bien que très utile, peut également être la source de mauvais choix d'outils. Les raisons sont multiples : avoir vu une présentation ou une démonstration sur un outil et vouloir refaire pareil, choisir un outil pour sa renommée ou plus simplement choisir un outil sans avoir bien cerné son besoin et son contexte.

Pour bien cibler son besoin, il est essentiel d'avoir identifié les raisons de l'automatisation :



Les raisons de l'automatisation

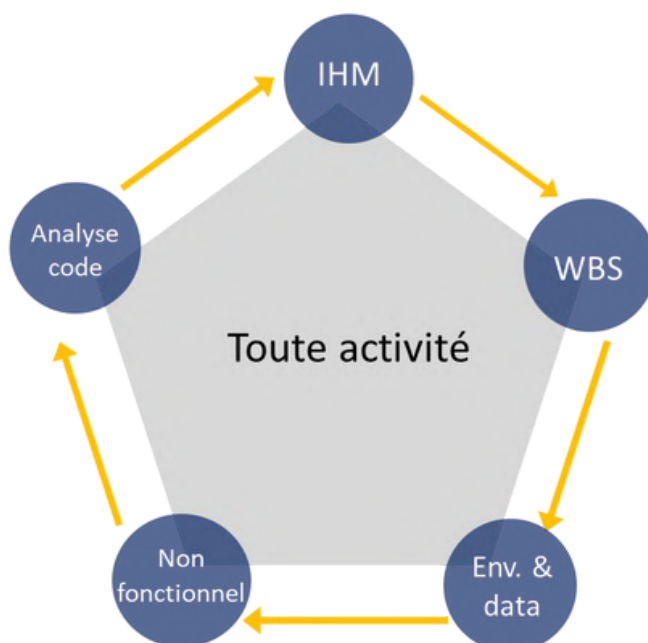


Source : Tutoriel JFTL 2022 : Choisir l'automatisation « pour tous »

Mais aussi ce qu'il est nécessaire d'automatiser :



Les périmètres de l'automatisation



Source : Tutoriel JFTL 2022 : Choisir l'automatisation « pour tous »

Bien identifier ce besoin permet de faire un premier tri dans les outils possibles.

Afin d'affiner ce tri, il est important de se pencher sur le contexte de cette automatisation. **Ce contexte va dépendre de critères comme :**

- Le budget disponible à investir (coût de licence, formation...).
- Le niveau de compétence des personnes en charge de l'automatisation (plus ou moins technique).
- L'environnement de développement...

Cet affinage va permettre de sélectionner un faible nombre d'outils répondant sur le papier à votre besoin et dans votre contexte. Il est alors nécessaire d'expérimenter ces outils et de faire un POC avec le(s) outil(s) sélectionné(s), qui permettra de voir si en condition réelle le(s) outil(s) répond à votre besoin. Et dans le cas où plusieurs outils répondent favorablement, lequel s'adapte le mieux. **Voici des exemples de critères qui reviennent souvent :**

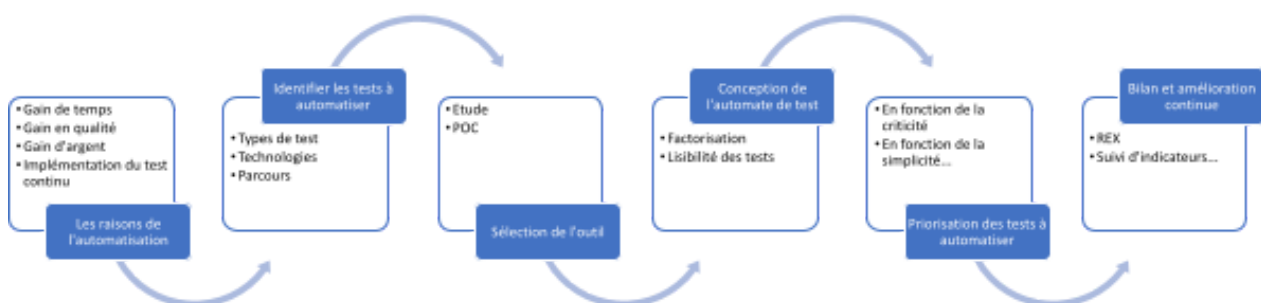
- La capacité de **factorisation** afin d'assurer une maintenance réduite des tests.
- La **fiabilité** afin d'éviter les pertes de temps suite à des tests en échec sans réelles raisons.
- Les **technologies** (web, mobile, desktop...) couvertes pour vérifier si l'outil s'adapte totalement à notre environnement technique.
- Le **niveau technique** requis pour s'assurer que tous les utilisateurs ciblés pourront manipuler efficacement l'outil.
- **L'ergonomie** pour faciliter la prise en main par des testeurs non-développeurs et donc l'utilisation effective de l'outil.
- La **réutilisation** ou **réversibilité** afin de ne pas perdre (ou perdre totalement) ses tests si on n'a plus la licence ou si l'éditeur disparaît.
- La **vitesse d'écriture des scripts**, car plus les tests sont rapides à écrire, plus ils pourront être exécutés rapidement.

- Les **coûts** en termes de licence, formation et infrastructures à mettre en place pour l'outil.
- Le **support** de l'éditeur ou de la communauté est primordial pour réussir à se débloquer rapidement lorsque l'on fait face à un problème.
- La **prise en main** et la **formation** nécessaire pour pouvoir utiliser correctement l'outil. Cet aspect est lié au coût, mais aussi à la capacité à mettre en place l'automatisation et à la montée en compétence des nouveaux utilisateurs.
- La **lisibilité** des tests, particulièrement si l'on souhaite partager ces tests avec des profils peu techniques.
- Le **retour sur investissement** (ROI), c'est-à-dire au bout de combien de temps/exécution il est pertinent financièrement d'automatiser ses tests.

Le POC doit fournir des réponses à vos diverses questions. Il se peut même qu'un POC débouche sur la non-sélection d'un outil et qu'un nouveau POC soit lancé sur des outils différents. Cela ne veut pas dire que le POC a échoué, bien au contraire, il a permis de ne pas trop investir dans un outil qui n'aurait pas satisfait vos besoins.

b. Comment élaborer votre stratégie d'automatisation des tests

Bien choisir un outil est essentiel. Cela n'est cependant qu'une partie de la stratégie d'automatisation des tests. Il est en effet nécessaire d'avoir une vision claire des livrables. Voici une stratégie standard liée à l'automatisation des tests :



Source : Tutoriel JFTL 2022 : Choisir l'automatisation « pour tous »

Les parties précédant le POC ont déjà été abordées dans cet ebook. Concentrons-nous maintenant sur celles d'après. Elles sont essentielles car une stratégie d'automatisation ne s'arrête pas au POC.

Suite au POC s'ouvre une phase d'industrialisation de l'automatisation. Lors de cette phase, il est généralement nécessaire de rattraper une dette d'automatisation parce que de nombreux tests existent déjà. Il est alors nécessaire de concevoir un automate qui permettra de maintenir et d'écrire facilement et rapidement ses tests. Pour cela, la factorisation et les bonnes pratiques de développement sont essentielles.

Au-delà de la construction de l'automate (plus ou moins chronophage en fonction de l'outil sélectionné), il est important de bien prioriser ses tests afin d'avoir des résultats rapides et utiles. Pour faire simple, c'est un peu comme gérer un backlog avec des tests qui ont plus ou moins de valeurs à être automatisés et demandent plus ou moins d'effort. Une bonne priorisation des tests se fait avec une bonne gestion de ces deux paramètres.

Enfin, quand cette phase d'industrialisation est en place, il est toujours important de dresser un bilan qui servira de REX à d'autres équipes. Il est aussi important de ne pas rester sur ses acquis et d'implémenter des démarches d'amélioration continue pour réviser ce qui a été mis en place.



A PROPOS D'AGILITEST

Faire de la validation continue une réalité

Pour devancer leurs concurrents, les entreprises doivent fournir à leurs clients des logiciels adaptés à leurs besoins et offrant une expérience client fluide et agréable. Cela signifie souvent :

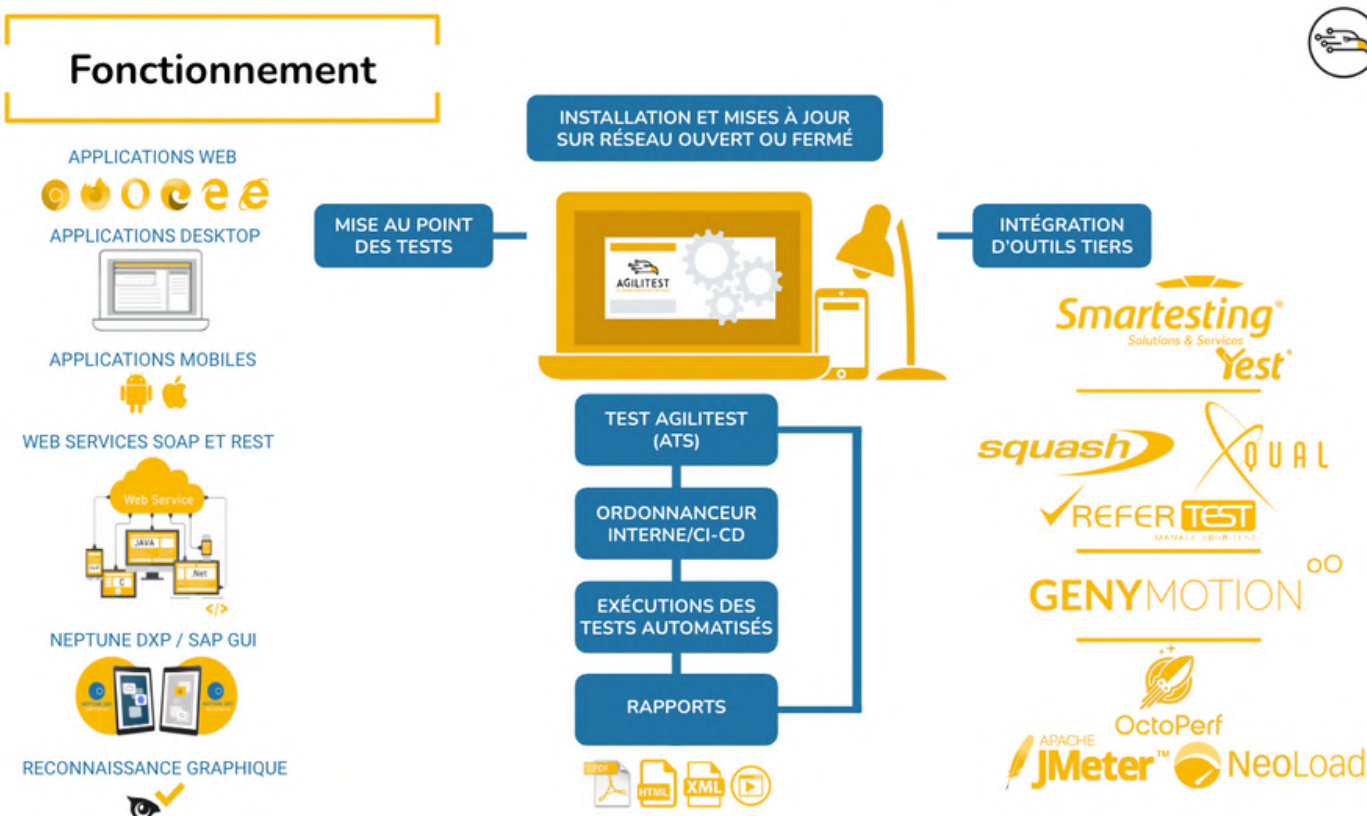
- Livrer le logiciel rapidement
- Livrer des corrections de bogues encore plus rapidement
- Sortir de nouvelles fonctionnalités fréquemment.

De plus, l'expérience de test doit être identique quelle que soit la technologie ou le canal testé et les tests en production ne doivent pas souffrir de contraintes techniques.

C'est pourquoi avoir une vue constante sur la qualité des logiciels est la clé pour permettre une véritable livraison continue. Cela permet aux équipes de se concentrer sur l'expérience client et la mise en œuvre des cas d'utilisation.

Agilitest propose une approche unique pour la conception de tests fonctionnels automatisés. À l'ère des équipes agiles et de l'intégration DevOps, Agilitest facilite la tâche des testeurs - tant techniques que fonctionnels - pour créer, maintenir et exécuter des tests fonctionnels automatisés robustes.

Les équipes de développement doivent sécuriser leurs processus tout en assurant une mise sur le marché plus rapide dans un secteur numérique hétérogène, multiplateforme et en constante évolution. Agilitest fournit aux QA Managers, aux Testeurs et aux Développeurs une plateforme de validation et un langage commun afin de valider efficacement leurs applications, d'avoir une visibilité globale et constante de la qualité des logiciels, et ainsi de rendre possible une approche de livraison continue.



Agilitest s'appuie sur 5 piliers :



ACCESSIBLE & NO-CODE

Agilitest est facile à utiliser pour les testeurs, puissant pour les développeurs mais surtout très performant pour tous.

Les experts fonctionnels n'ont plus besoin de compétences en codage ou de l'aide de développeurs pour créer facilement des stratégies de tests automatisés et exécuter eux-mêmes les tests. L'**interface no-code** d'Agilitest a été conçue dans un souci d'efficacité et offre aux utilisateurs une expérience fluide. **La création de tests fonctionnels automatisés n'a jamais été aussi intuitive.**



FACILE À MAINTENIR

75% des coûts d'automatisation des tests sont consacrés à la maintenance des tests. Cela doit changer. Nous nous sommes débarrassés des Xpaths et avons créé un moyen beaucoup plus robuste d'identifier les composants utilisés pour exécuter le test. Ce **système de capture innovant** et puissant rend les tests moins sensibles aux modifications du code logiciel.

Oubliez les tests défaillants et détectez les régressions grâce à **des rapports PDF, XML et vidéo exploitables**. Lorsqu'un test échoue, il est essentiel de pouvoir identifier rapidement et efficacement la raison de cet échec. Agilitest génère automatiquement **des reprises de test en vidéo** ainsi que des rapports XML et PDF contenant des informations clés sur les résultats du test, les conditions et les captures d'écran afin que vous puissiez déterminer rapidement si le scénario de test doit être maintenu ou si vous êtes confronté à une régression.



OPEN SOURCE

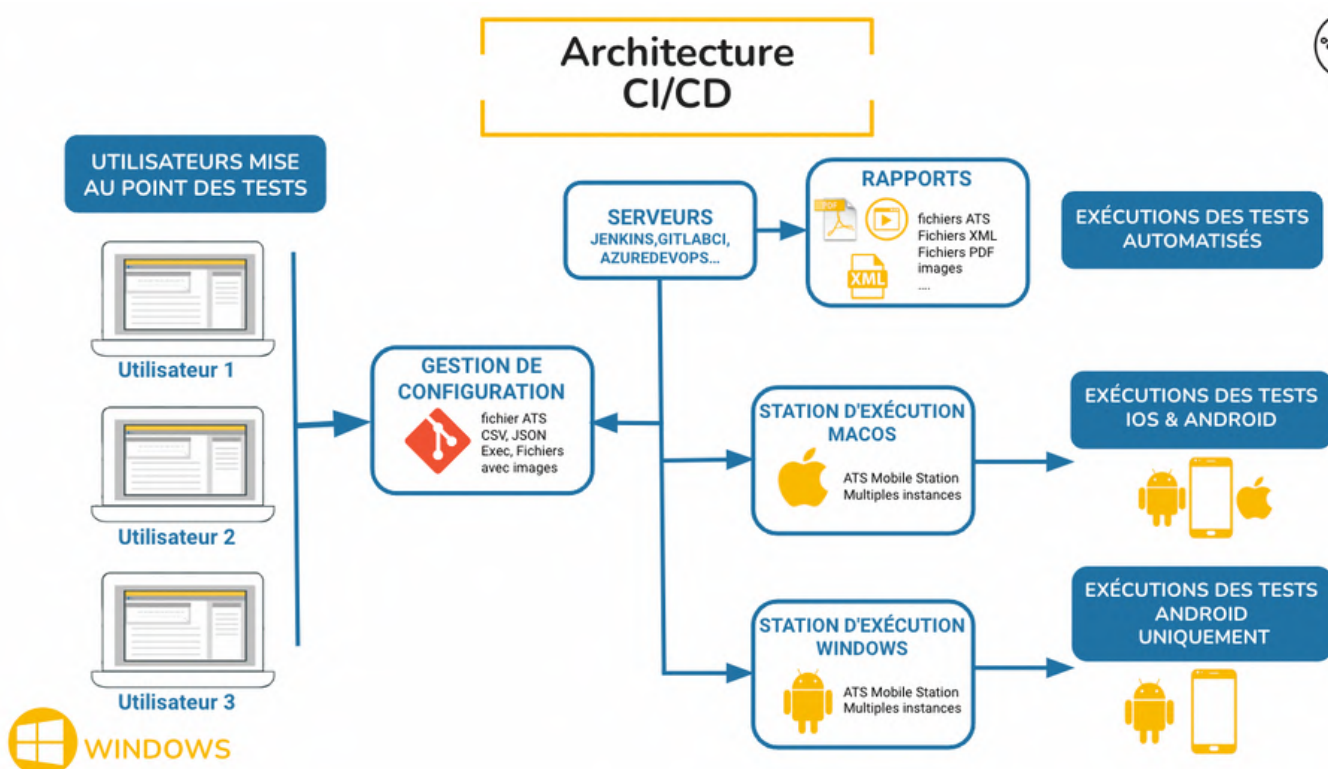
Nous vous aidons à vous débarrasser des obstacles liés aux fournisseurs de tests. Agilitest produit des **scripts au format ActionTestScript (ATS)**, qui sont lisibles et modifiables manuellement. Le format ATS est Open Source : les scripts ATS peuvent être **rejoués librement** dans Agilitest ou sur tout système d'intégration continue supportant Java/TestNG.



FACILE À DÉPLOYER

Agilitest est **multi-canal** et vous permet de créer et d'exécuter une large gamme de scénarios, vous permettant de réexécuter le même test dans plusieurs environnements, mais aussi de produire des scénarios réalistes **multi-applications, multi-technologies** et de les tester. Intégrez toutes vos exécutions de test avec vos outils CI préférés pour déclencher automatiquement des exécutions de test une fois que votre build est prêt.

Le format ATS d'Agilitest permet **une compatibilité maximale** avec les systèmes les plus utilisés (Jenkins, Maven, TestNG, etc). Enfin, nos rapports et vidéos commentés vous permettent de rejouer les exécutions de test et de comprendre où et pourquoi il a échoué.



SUPPORT PREMIUM

Agilitest fournit un support premium en ligne inclus dans toutes les licences. Nous fournissons un canal Slack personnel pour bénéficier d'un soutien direct et réactif de la part de nos consultants internes et de nos équipes R&D. Nous résolvons les problèmes quotidiens lorsqu'ils surviennent. Il n'y a pas de questions ou de bugs laissés de côté et nous fournissons régulièrement des évolutions et des corrections.

L'approche unique d'Agilitest offre des gains significatifs pour les projets, une tranquillité d'esprit pour les équipes et fait de la validation continue une réalité.

Démarrez gratuitement.

Les 30 premiers jours sont offerts. Tous les scénarios de test que vous créez peuvent être rejoués gratuitement et pour toujours avec ATS, notre référentiel open source.

Démarrer un essai gratuit

www.agilitest.com

**Ils font confiance à Agilitest pour l'automatisation
des tests logiciels**

CANAL+

L'ORÉAL

GAMR FIRST
CASINO BARRIERE MONTREUX

Dedalus
HEALTHCARE SYSTEMS GROUP

