Engineer

# Tests de régression

tests qui vérifient les acquis après un changement

## Ce que c’est

Les tests de régression vérifient qu’après un changement, un logiciel conserve des comportements qui fonctionnaient déjà. Une option d’export peut respecter son nouveau cahier des charges tout en modifiant les lignes que reçoivent les utilisateurs actuels. Tester cette seule option indique si elle fonctionne.
Le test de régression emprunte aussi l’ancien parcours pour vérifier qu’un comportement sur lequel des personnes ou des intégrations comptent est préservé.

Le manuel de génie logiciel de la NASA présente ces tests comme un moyen de vérifier qu’une modification n’a pas introduit de défaut. Il propose notamment de relancer des tests rattachés au code ou aux exigences modifiées et de prendre en compte les modules qui interagissent. L’idée s’applique aux changements de code source, de dépendance, de configuration, de données ou de milieu d’exécution. Le même programme peut répondre autrement après un changement de navigateur ou d’entrée.

Confirmer une correction et chercher des régressions sont deux démarches complémentaires. On reproduit le défaut, puis on constate le résultat corrigé. On contrôle ensuite les comportements voisins qui doivent rester valides. Le cas du défaut initial peut être conservé pour signaler son retour. Le résultat attendu doit refléter un engagement du produit. Si les exigences changent volontairement, actualisons cet attendu et décrivons ce que les applications qui l’appellent devront faire différemment.

Utilisez le concept pour prendre une décision

Utilisez « Que sont les tests de régression ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Une seule équipe transforme une idée en un produit auquel vos clients peuvent se connecter, sans le faire passer entre trois prestataires.

**Découvrir nos prestations** Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Application GIS : revenus multipliés par 10 — 10x croissance du chiffre d'affaires portée par le produit cartographique

**Application GIS : revenus multipliés par 10** Contactez-nous →

## Pourquoi c’est important

Les conséquences d’un changement dépassent souvent les lignes modifiées. Transformer une liste en ensemble peut supprimer les doublons et modifier l’ordre d’un rapport. Un analyseur de dates peut comprendre un format nouveau et perdre la prise en charge d’un champ vide en amont. Un test automatisé aide à rattacher un effet à un changement précis. Cela compte surtout lorsque le comportement sert une intégration rarement lancée sur le poste de développement.

Pour une petite modification, partons de la règle partagée la plus proche et d’un ou deux appelants réels différents. Vérifions les effets possibles sur la configuration par défaut, les limites, les erreurs ou les données déjà conservées. Lorsque le changement est important, élargissons les observations à des propriétés comme la performance et l’accessibilité. Le niveau de preuve dépend de la conséquence d’un défaut manqué. Un test unitaire peut suffire pour une conversion. Une vérification dans un navigateur peut être nécessaire pour examiner l’affichage ou la navigation au clavier véritables.

Un exemple nommé explique la raison d’une protection. « Une date manquante reste vide lors de l’export d’anciens dossiers » renseigne davantage que « le cas trois passe ». Présentons les entrées utiles et le résultat correspondant. La personne qui examinera un changement pourra savoir si une nouvelle exigence modifie volontairement cet attendu ou si une fonction déjà utilisée a été cassée.

## À quoi cela ressemble en pratique

Prenons un export CSV qui reçoit une option pour inclure des comptes archivés. Par défaut, ces comptes restent exclus. Les comptes actifs restent triés par référence, et la date du dernier contact reste vide lorsqu’elle manque. La

nouvelle option doit ajouter les archives sans dupliquer les lignes actives. Préparons deux comptes actifs dans l’ordre inverse et un compte archivé. La date manque dans une ligne. Exerçons les deux parcours par l’exporteur réel.

Sur le parcours par défaut, confrontons les références, l’ordre et la cellule sans date. Avec l’option, vérifions que les comptes constatés précédemment et le compte archivé apparaissent chacun une fois. Ajoutons un nom contenant une virgule. Un exporteur qui concatène du texte peut produire le bon nombre de lignes tout en décalant les colonnes. Lisons le CSV avec un outil comparable à celui du client. Le jeu de données doit rester compréhensible tout en donnant du relief au comportement.

Vérifions ensuite que les tests rejettent une vraie régression. Faisons temporairement apparaître les comptes archivés par défaut ou retirons les guillemets autour du champ avec virgule. Les contrôles correspondants doivent échouer pour des raisons distinctes et lisibles. Restaurons le code et rejouons les tests. Une assertion qui vérifie simplement l’existence du fichier pourrait laisser passer les deux défauts. Une entrée sans archive protège également le cas valable où la sélection est vide.

Notre note sur les soulignements de Safari et les sous-ensembles de police illustre une autre limite. Une modification CSS paraissait résoudre le problème de Chromium. Safari continuait de tracer le soulignement à travers les jambages descendants. Une suite de régression de ce comportement doit réellement observer les moteurs et les cas qu’elle prétend couvrir. La note décrit le diagnostic et ses observations. Le test proposé prolongerait cette leçon.

## Où les problèmes surviennent

Le nombre de tests peut donner une impression trompeuse de couverture. Des centaines d’assertions élémentaires peuvent manquer le seul appelant qui lise les données historiques. Choisissons les exemples à partir des usages, des entrées et des interactions réellement affectées. Quand un défaut a déjà échappé aux vérifications, protégeons le parcours touché et vérifions que l’assertion échoue si le défaut réapparaît.

On peut aussi comparer trop de choses. Une capture intégrale évolue quand l’ordre, les identifiants aléatoires ou l’heure changent. Vérifions l’ordre si l’export s’y engage. S’il garantit seulement un ensemble d’enregistrements, comparons

l'ensemble en vérifiant aussi l'absence de doublons. Un attendu modifié volontairement mérite une revue comparable au changement de code et une raison conservée.

Enfin, un test instable rend les échecs difficiles à interpréter. Attendons une condition observable avec une durée maximale, fixons des données connues et évitons de laisser de l'état entre les essais. Gardons les vérifications larges aux endroits où les vraies interactions comptent et les essais ciblés près du travail quotidien. Ils apportent des preuves à des niveaux différents.

## Pour aller plus loin

- Manuel de génie logiciel de la NASA sur les tests de régression
- Documentation de pytest sur la paramétrisation des cas
- Tests unitaires dans Software Engineering at Google

Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

**Consulter les tarifs** Contactez-nous →

# Rendez la terminologie concrète

Demandez comment « Que sont les tests de régression ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/regression-testing/