Engineer

Défaut

imperfection rendant un produit non conforme à ses exigences

Ce que c’est

Un défaut est une imperfection dans un travail réalisé, qui entraîne un comportement différent de celui prévu ou empêche d’accomplir une partie de ce qui était attendu. La plupart des programmeurs l’appellent un bug ; les testeurs et les normes emploient aussi les mots anglais fault ou flaw pour désigner cette imperfection.

Le mot important est ici *travail*. Un défaut peut résider dans le code source, mais tout autant dans une exigence qui en contredit une autre, une conception incapable de supporter la charge pour laquelle elle a été prévue, un fichier de configuration, un test dont la valeur attendue est incorrecte ou un manuel décrivant un bouton supprimé deux versions auparavant. Où qu’il réside, le défaut appartient à l’élément lui-même et existe, que quelqu’un l’ait remarqué ou non.

Le vocabulaire des tests distingue trois mots voisins, et cette distinction est utile. Une personne commet une erreur : une spécification mal lue, une hypothèse fausse, une faute de frappe. Cette erreur laisse un défaut dans ce que la personne a produit. Lorsque la partie défectueuse s’exécute dans des conditions qui exposent ce défaut, le produit se comporte mal : ce comportement erroné visible est une défaillance. La chaîne va dans un sens et peut s’arrêter à n’importe quel maillon. Un défaut dans une branche jamais exécutée, ou qui exige une entrée que personne n’envoie, peut rester des années dans une base de code sans provoquer une seule défaillance.

Ce qui est erroné dépend de ce qui était attendu, et les attentes vont au-delà de la spécification écrite. Le modèle de qualité des produits d’ISO/IEC 25010 définit l’adéquation fonctionnelle comme le degré auquel un produit « fournit des fonctions qui répondent aux besoins explicites et implicites lorsqu’il est utilisé

dans des conditions spécifiées». Il la décompose en complétude (les fonctions couvrent-elles les tâches spécifiées et les objectifs des utilisateurs visés?), en exactitude (les résultats sont-ils exacts?) et en pertinence (les fonctions aident-elles à accomplir la tâche?). Un total faux relève de l'exactitude; un rapport que personne ne peut exporter, de la complétude. Le même modèle traite la fiabilité comme une caractéristique distincte comprenant notamment l'absence de défauts, la disponibilité, la tolérance aux pannes et la capacité de récupération. Un produit qui calcule la bonne réponse puis plante deux fois par jour présente donc encore des défauts.

Une régression est un défaut qui a une histoire : quelque chose fonctionnait et a été cassé par une modification ultérieure. Elle mérite son nom propre parce qu'elle apporte un indice absent d'un défaut initial : une dernière version dont on sait qu'elle fonctionnait.

Utilisez le concept pour prendre une décision

Utilisez «Qu'est-ce qu'un défaut?» 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

Le coût d'un défaut dépend surtout de l'endroit où il est découvert. Repéré par la personne qui écrit le code, il coûte une minute et ne laisse aucune trace. Repéré en revue, il coûte une conversation. Repéré par un client, il coûte la défaillance elle-même, le ticket d'assistance, l'enquête, la correction et un peu de confiance; s'il a causé un préjudice réel, il est devenu un élément d'un incident. Systèmes de types, linters, tests et revues de code suivent la même logique : rapprocher le moment de la découverte de celui où le défaut a été écrit.

Les défauts ont aussi des familles. L'erreur à l'origine de l'un a généralement des parentes : la même interface mal comprise appelée de la même manière à six endroits, la même confusion d'unités copiée dans chaque nouveau module, la même hypothèse non vérifiée dans chaque gestionnaire de requête. Après

la découverte d'un défaut, la question productive est moins de savoir où est la ligne cassée que ce qui l'a rendue facile à écrire et où le même problème s'est reproduit.

Et les défauts reviennent. Le code change constamment, et un défaut corrigé peut revenir à la faveur d'une refactorisation, d'une fusion ou d'une mise à niveau de dépendance. Un test conçu pour échouer sur le défaut et réussir sur la correction fait de celle-ci une propriété durable du processus de construction. Quand une régression apparaît malgré tout, l'historique des versions devient un espace de recherche : `git bisect` effectue une recherche dichotomique dans les commits entre une version connue comme correcte et une version connue comme défectueuse. Un intervalle de mille commits se réduit ainsi à celui qui a introduit le bug en environ dix exécutions de test.

Certains défauts sont aussi des problèmes de sécurité. La Common Weakness Enumeration de MITRE répertorie des faiblesses, définies comme « une condition dans un composant logiciel, micrologiciel, matériel ou de service qui, dans certaines circonstances, pourrait contribuer à l'introduction de vulnérabilités ». Beaucoup d'entrées sont des défauts ordinaires qu'un attaquant peut exploiter : vérification de limites absente, entrée utilisateur transmise sans échappement à une requête, étape d'autorisation omise sur une route.

À quoi cela ressemble en pratique

Les défauts les plus coûteux sur ce site étaient ceux pour lesquels tous les indicateurs automatiques étaient au vert. Un audit a révélé onze défauts d'accessibilité que les règles automatisées avaient laissés passer, parce que chacun relevait du rendu d'une page plutôt que de ce que disait son balisage. Deux se trouvaient dans les contrôles eux-mêmes. Toutes les fenêtres ouvertes par le contrôle de mise en page mesuraient 900 pixels de haut : un téléphone à l'horizontale, de 852 sur 393 pixels, avec sept pixels entre deux éléments, n'avait donc jamais été examiné. Et le moteur de règles n'exécutait que 70 de ses 105 règles. Chaque défaut a rejoint un registre avec la règle qu'il avait suscitée, afin de traiter la catégorie d'erreurs plutôt que sa seule occurrence. Une fois le contrôle de mise en page lui-même réparé, il a échoué 128 fois dans trois langues.

Un second défaut ne produisait aucun symptôme. Le site propose par défaut un thème sombre et un thème clair pour les lecteurs dont le système le demande ; sa couverture pleine page comporte trois couches dans le thème sombre. Un

petit contrôle structurel comptant les couches peintes par chaque thème a constaté que le thème clair n'en dessinait que deux dès sa première exécution. La surcouche manquait depuis l'écriture du thème clair. Aucune erreur n'était levée, toutes les pages s'affichaient, et un arrière-plan avec une couche en moins pouvait passer pour un choix délibéré. Pendant des mois, rien n'incitait donc à regarder. La correction tenait en une règle ; le contrôle est ce qui la maintient.

Maintenir les corrections est le rôle du moteur de garde-fous du dépôt, où une règle lie chaque correction de bug à une preuve : un commit marqué comme correctif doit apporter un test qui échoue sans lui. Les garde-fous ont aussi leurs propres tests, au nombre de 146, car un garde-fou défectueux peut laisser passer les changements qu'il devait rejeter tout en donnant l'impression de fonctionner.

Où les problèmes surviennent

L'occurrence est corrigée, mais pas la famille d'erreurs. La ligne signalée est réparée, le ticket est fermé et trois erreurs de la même famille attendent ailleurs dans le code. Après chaque correction, recherchez le motif à l'origine du défaut ; si une machine peut le reconnaître, transformez cette recherche en contrôle.

Un bug corrigé revient. Une refactorisation six mois plus tard rétablit l'ancien comportement, car rien ne consignait ce que protégeait la correction. Écrivez le test avant le correctif, observez son échec sur l'ancien code et conservez-le définitivement dans la suite.

Le vérificateur lui-même contient le défaut. Un ensemble de règles qui n'en charge silencieusement qu'une fraction, une liste de fenêtres de test ayant toutes la même hauteur, une assertion vérifiant ce que fait le code plutôt que ce qu'il devrait faire : chacun annonce un succès alors que le défaut est bien visible. Donnez aux contrôles leurs propres tests et lisez dans la sortie de l'outil quelles règles se sont réellement exécutées.

La couverture remplace l'exactitude. Une suite peut exécuter chaque ligne et tout de même accepter un résultat faux si aucune assertion ne l'examine. Un code conforme à sa spécification écrite peut aussi ne pas répondre à un besoin implicite. Utilisez la couverture pour trouver le code que rien ne teste et jugez un test d'après les défauts qu'il détecterait.

« Conforme à la conception » clôt un vrai signalement. Le comportement respecte la spécification, le signalement est rejeté et c'était la spécification elle-même qui était défectueuse depuis le début. Si le problème réside dans l'exigence ou la conception, rattachez-lui le défaut pour que la correction ait lieu à la source.

Pour aller plus loin

- Défaut — la définition ISTQB utilisée dans les tests logiciels, ses synonymes et les termes voisins erreur et défaillance.
- ISO/IEC 25010 — une présentation du modèle de qualité des produits et des sous-caractéristiques de chaque caractéristique, dont l'adéquation fonctionnelle et la fiabilité.
- git-bisect — le manuel Git de la recherche dichotomique du commit ayant introduit un bug, y compris les exécutions automatisées avec un script de test.
- À propos de CWE — ce qu'est la Common Weakness Enumeration, l'organisation de sa liste de faiblesses logicielles et matérielles et l'organisme qui la gère.

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 « Qu'est-ce qu'un défaut ? » 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/defect/