UNIVERSITÉ DE YAOUNDÉ 1
DÉPARTEMENT INFORMATIQUE

INFORMATION AND COMMUNICATION TECHNOLOGIES FOR DEVELOPMENT

ICT304

SOFTWARE TESTING AND DEPLOYMENT

BLACK BOX TESTING TECHNICS

Réalisé par :

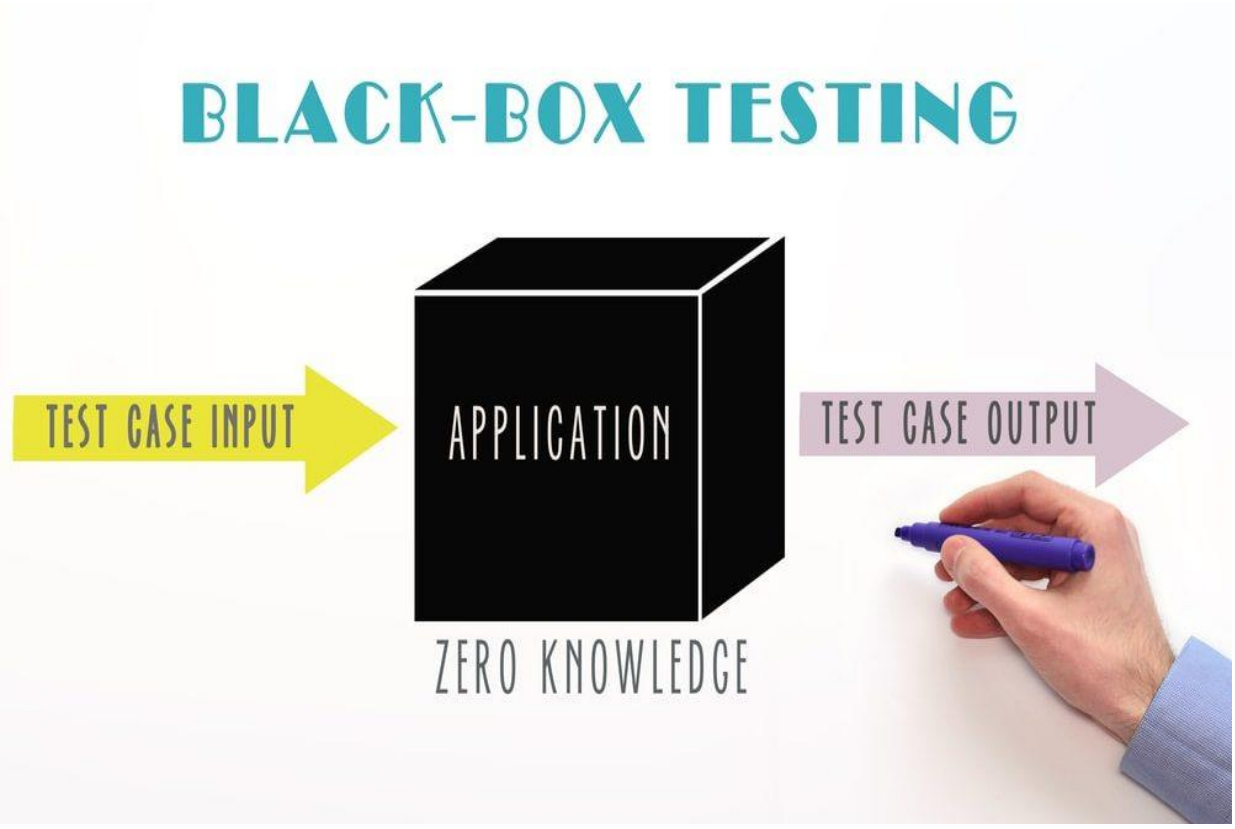
- MBIADA BAYON IDRIS DONALD 21Q2915
- NYADJOU LUCIE DANIELLE 21Q2301
- KAMELA PIERRICK DACK 21Q2493
- KEWOU MBOTCHAK CHRISTIAN 21Q2402
- NDOKOU ELAT DESIRE SAMIRA 21T2314

Année académique 2023-2024

Sommaire

I. INTRODUCTION

Le test de boîte noire est une méthode de test logiciel où le fonctionnement interne du système n'est pas pris en compte lors de la création des tests. Au lieu de cela, les tests se concentrent sur les entrées et les sorties du logiciel, évaluant son comportement sans connaissance préalable de sa structure interne. Cette approche est souvent comparée au test de boîte blanche, où les tests sont basés sur une compréhension détaillée du code source et de la logique interne du programme. Aussi, ce test consiste à évaluer le code, les entrées et les sorties du programme. Le test est choisi à partir des spécifications ou fonctionnalités du client ; ce test permet d'assurer l'adéquation entre la spécification et le code ; toutefois il est aveugle aux défauts de la programmation.



II. IMPORTANCE DU TEST BOITE NOIR

Le test de boîte noire revêt une importance capitale dans le processus de développement logiciel pour plusieurs raisons :

1. Indépendance par rapport à l'implémentation

Les tests de boîte noire permettent aux testeurs de se concentrer sur le comportement externe du logiciel, indépendamment de la manière dont il est implémenté. Cela facilite la détection des erreurs et des bogues sans être limité par la structure interne du code.

2. Simulation des perspectives de l'utilisateur final

En se basant uniquement sur les entrées et les sorties, le test de boîte noire offre une perspective similaire à celle de l'utilisateur final. Cela permet de s'assurer que le logiciel répond correctement aux besoins et attentes des utilisateurs.

3. Découverte des défauts de conception

Les tests de boîte noire peuvent révéler des erreurs de conception, telles que des fonctionnalités manquantes ou des interfaces utilisateur peu conviviales. En se concentrant sur le comportement global du logiciel, ces défauts peuvent être détectés et corrigés plus tôt dans le processus de développement.

4. Couverture plus large des cas d'utilisation

Étant donné que les tests de boîte noire ne dépendent pas de la connaissance du code source, ils peuvent être utilisés pour tester une grande variété de scénarios d'utilisation, y compris ceux qui n'ont pas été anticipés par les développeurs.

III. TECHNIQUES DU TEST BOITE NOIR

1. Tests d'équivalence

Les tests d'équivalence sont une technique de test de boîte noire qui vise à réduire le nombre de cas de test en regroupant les entrées en classes équivalentes. L'idée est que si une entrée de chaque classe fonctionne correctement, alors toutes les entrées de cette classe devraient également fonctionner correctement. Cela permet de réduire la redondance des tests et d'optimiser l'efficacité du processus de test.

➔ **Explication de la méthode** :

Pour appliquer les tests d'équivalence, les entrées sont divisées en classes équivalentes, où chaque classe représente un ensemble d'entrées qui devraient produire des résultats similaires lorsqu'elles sont traitées par le logiciel. Ensuite, un seul cas de test est sélectionné à partir de chaque classe pour être exécuté. Si le test réussi pour un cas donné, on suppose qu'il réussira pour tous les cas de cette classe.

Par exemple, supposons que nous devions tester une fonction qui accepte des entiers et renvoie vrai si l'entier est supérieur à 10. Les classes équivalentes pourraient être définies comme suit :

- Classe d'entiers inférieurs ou égaux à 10.
- Classe d'entiers supérieurs à 10.

Dans ce cas, un seul test serait sélectionné dans chaque classe pour être exécuté, par exemple :

- Test avec un entier de la classe inférieure ou égal à 10 (par exemple, 5).
- Test avec un entier de la classe supérieure à 10 (par exemple, 15).

➔ **Exemple illustratif** :

Prenons un exemple concret d'une application de gestion de magasin en ligne. Supposons que nous devions tester une fonctionnalité qui calcule le montant total d'une commande en tenant compte des frais de livraison.

Les classes équivalentes pourraient être :

- Classe d'articles avec livraison standard.
- Classe d'articles avec livraison express.

Dans ce cas, un seul test serait sélectionné dans chaque classe pour être exécuté :

- Test avec des articles de la classe de livraison standard.
- Test avec des articles de la classe de livraison express.

➔ **Avantages et limitations**

★ **Avantages :**

- **Réduction du nombre de cas de test :** Les tests d'équivalence permettent de réduire le nombre de tests nécessaires en regroupant les entrées en classes équivalentes.
- **Meilleure gestion des tests :** En réduisant la redondance des tests, cette méthode permet une gestion plus efficace des cas de test et des ressources nécessaires pour les exécuter.

★ **Limitations :**

- **Simplification excessive :** La méthode des tests d'équivalence peut parfois simplifier excessivement la complexité du système, en regroupant des entrées qui peuvent avoir des comportements différents mais qui tombent dans la même classe.

- **Négligence des frontières :** Cette méthode peut négliger les frontières entre les classes équivalentes, où le comportement du système peut changer. Cela peut conduire à des cas de test manquants pour des situations critiques.

2. Tests de valeurs limites

Les tests de valeurs limites sont une technique de test de boîte noire qui se concentre sur les valeurs limites des données d'entrée pour identifier les erreurs potentielles dans le traitement des valeurs aux limites. Cette méthode est particulièrement efficace pour détecter des erreurs telles que les débordements, les erreurs de condition aux limites et les problèmes de traitement des bords des données.

➔ Explication de la méthode

Les tests de valeurs limites consistent à sélectionner les valeurs justes à l'intérieur et juste à l'extérieur des limites des plages valides d'entrées. L'objectif est de tester le comportement du logiciel aux extrémités de ces plages, où les erreurs sont plus susceptibles de se produire.

Par exemple, si une fonctionnalité du logiciel accepte des valeurs entre 1 et 100, les tests de valeurs limites pourraient inclure les valeurs 0, 1, 100 et 101 pour évaluer le comportement du logiciel dans ces situations limites.

➔ Exemple illustratif

Prenons l'exemple d'un système de réservation en ligne qui permet aux utilisateurs de choisir le nombre de billets pour un événement. Si le système spécifie qu'un utilisateur peut réserver entre 1 et 10 billets, les tests de valeurs limites pourraient inclure :

- 0 billet : juste en dessous de la limite inférieure.
- 1 billet : à la limite inférieure.
- 10 billets : à la limite supérieure.
- 11 billets : juste au-dessus de la limite supérieure.

En testant ces valeurs, on peut détecter des erreurs potentielles telles que des messages d'erreur incorrects, des débordements ou des problèmes de traitement des cas limités.

➔ **Avantages et limitations**

★ **Avantages**

- **Détection précoce des erreurs :** Les tests de valeurs limites permettent de détecter des erreurs potentielles aux limites des plages valides d'entrées, ce qui peut contribuer à une détection précoce des problèmes dans le logiciel.
- **Couverture des cas limites :** En se concentrant sur les valeurs aux limites, cette méthode garantit une couverture adéquate des scénarios critiques où les erreurs sont plus susceptibles de se produire.

★ **Limitations**

- **Limitation de la portée :** Les tests de valeurs limites ne garantissent pas la détection de toutes les erreurs potentielles dans le logiciel. Ils se concentrent uniquement sur les valeurs aux limites et peuvent manquer d'erreurs dans d'autres parties du système.
- **Complexité accrue :** La gestion des tests de valeurs limites peut être plus complexe, car elle nécessite une identification précise des limites des plages valides d'entrées et la sélection appropriée des valeurs de test.

3. Tests de combinaison

Les tests de combinaison, également appelés tests de cas de test combinatoires, sont une technique de test de boîte noire qui consiste à tester différentes combinaisons d'entrées pour identifier les interactions entre celles-ci. Cette méthode vise à évaluer le

comportement du système lorsque plusieurs variables sont combinées, ce qui permet de détecter des erreurs qui pourraient ne pas être détectées lors de tests individuels.

➔ Explication de la méthode

Dans les tests de combinaison, les valeurs possibles pour chaque entrée sont combinées de manière à couvrir un ensemble représentatif de scénarios d'utilisation. Les combinaisons peuvent être générées manuellement ou à l'aide de techniques automatiques, telles que la génération de combinaisons basée sur la couverture de pair.

Par exemple, si un logiciel de réservation d'hôtel prend en compte des variables telles que le type de chambre, la durée du séjour et les options de restauration, les tests de combinaison pourraient inclure des combinaisons de ces variables pour évaluer le comportement du système dans différentes situations.

➔ Exemple illustratif

Supposons que nous devons tester un système de planification de voyage en ligne qui permet aux utilisateurs de choisir leur destination, les dates de départ et de retour, ainsi que les options de transport (avion, train, voiture).

Les tests de combinaison pourraient inclure des combinaisons telles que :

- Destination : Paris, Dates : 10/05/2024 - 15/05/2024, Transport : Avion.
- Destination : New York, Dates : 15/06/2024 - 20/06/2024, Transport : Train.
- Destination : Tokyo, Dates : 01/07/2024 - 10/07/2024, Transport : Voiture.

En testant ces combinaisons, on peut identifier des erreurs telles que des conflits de dates, des incompatibilités de transport ou des problèmes de disponibilité dans certaines destinations.

➔ **Avantages et limitations**

★ **Avantages**

- **Détection des interactions entre variables :** Les tests de combinaison permettent de détecter des erreurs qui résultent de l'interaction entre différentes variables, ce qui peut être difficile à identifier lors de tests individuels.
- **Couverture exhaustive des scénarios :** En testant plusieurs combinaisons de valeurs pour différentes entrées, cette méthode offre une couverture plus exhaustive des scénarios d'utilisation du système.

★ **Limitations**

- **Complexité accrue :** La génération et la gestion des tests de combinaison peuvent être plus complexes en raison du nombre potentiellement élevé de combinaisons à tester, ce qui peut augmenter les efforts nécessaires pour effectuer les tests.
- **Surcharge de tests :** Tester toutes les combinaisons possibles peut entraîner une surcharge de tests, ce qui peut rendre le processus de test inefficace et coûteux en termes de temps et de ressources.

4. Tests aléatoires

Les tests aléatoires, également connus sous le nom de tests aléatoires ou de tests basés sur la génération de données aléatoires, sont une technique de test de boîte noire où les entrées de test sont générées de manière aléatoire sans suivre de schéma préétabli. Cette

méthode vise à explorer différentes parties du logiciel de manière aléatoire pour détecter des erreurs potentielles qui pourraient ne pas être découvertes par d'autres techniques de test plus ciblées.

➔ **Explication de la méthode**

Dans les tests aléatoires, les entrées de test sont générées de manière aléatoire à partir d'un ensemble d'options possibles. Les valeurs aléatoires sont sélectionnées pour chaque entrée de test, et les tests sont exécutés pour évaluer le comportement du système. Cette approche permet d'explorer un large éventail de scénarios d'utilisation sans avoir besoin de spécifier des cas de test individuels.

➔ **Exemple illustratif**

Prenons l'exemple d'un système de gestion de bibliothèque en ligne. Pour effectuer des tests aléatoires, nous pourrions générer aléatoirement des scénarios tels que :

- Ajouter un nouveau livre avec un titre et un auteur généré aléatoirement.
- Rechercher un livre en utilisant des termes de recherche aléatoires.
- Modifier les détails d'un livre, tels que le titre, l'auteur ou la description, avec des valeurs aléatoires.
- Supprimer un livre de la bibliothèque en sélectionnant aléatoirement un livre existant.

En exécutant ces tests aléatoires, on peut explorer différentes parties du système et identifier des erreurs potentielles dans son comportement.

➔ **Avantages et limitations**

★ **Avantages**

- **Exploration exhaustive :** Les tests aléatoires permettent d'explorer un large éventail de scénarios d'utilisation, ce qui peut révéler des erreurs qui auraient pu être manquées par d'autres techniques de test plus ciblées.
- **Découverte de bugs inattendus :** En générant des entrées de test de manière aléatoire, cette méthode peut mettre en évidence des erreurs inattendues ou des comportements inattendus du système.

★ **Limitations**

- **Manque de reproductibilité :** Les tests aléatoires peuvent manquer de reproductibilité car les résultats peuvent varier d'une exécution à l'autre en raison de la nature aléatoire de la génération des données de test.
- **Faible couverture de certaines parties du système :** Les tests aléatoires peuvent ne pas couvrir de manière adéquate certaines parties du système, en particulier les cas limites ou les situations spécifiques qui nécessitent une configuration précise.

IV. APPLICATION DU TEST BOITE NOIR

Les techniques de test de boîte noire sont largement utilisées dans l'industrie du logiciel pour garantir la qualité et la fiabilité des produits logiciels. Voici quelques exemples de cas réels où ces techniques ont été utilisées avec succès pour identifier des bugs :

1. Application de messagerie instantanée

Une entreprise développe une application de messagerie instantanée pour les appareils mobiles. Lors des tests de boîte noire, les testeurs ont utilisé des scénarios de tests de combinaison pour simuler différents cas d'utilisation, tels que l'envoi de messages texte, d'images et de fichiers, avec des combinaisons de différentes tailles de fichier et de réseaux (Wi-Fi, données mobiles). Cette approche a permis de détecter un bogue où l'application plantait lors de l'envoi de fichiers volumineux via une connexion réseau mobile instable.

2. Plateforme de commerce électronique

Une société de commerce électronique met à jour sa plateforme de vente en ligne pour inclure une nouvelle fonctionnalité de recommandation de produits. Les tests de boîte noire ont été utilisés pour vérifier la fonctionnalité en utilisant des scénarios de tests d'équivalence et de valeurs limites pour différents types de produits, montants de commandes et historiques d'achat. Les testeurs ont découvert un bogue où les recommandations de produits n'étaient pas mises à jour correctement après la modification du panier d'achat, ce qui aurait pu entraîner des recommandations inappropriées pour les clients

3. Application de gestion des ressources humaines

Une entreprise déploie une application de gestion des ressources humaines pour gérer les demandes de congé des employés. Lors des tests de boîte noire, les testeurs ont utilisé des tests de scénarios pour simuler différents cas, tels que la demande de congé, l'approbation par les supérieurs hiérarchiques et le suivi des soldes de congés. Une utilisation intensive des tests de combinaison a été effectuée pour vérifier les interactions entre les différents types de congés (maladie, vacances, congé de maternité) et les politiques de l'entreprise. Cela a permis de découvrir un bogue où

les congés de maternité n'étaient pas correctement pris en compte dans le calcul des soldes de congés.

Ces exemples démontrent l'efficacité des techniques de test de boîte noire dans la détection de bugs et d'anomalies dans divers types de logiciels. En combinant différentes approches telles que les tests d'équivalence, les tests de valeurs limites, les tests de combinaison et les tests de scénarios, les testeurs peuvent garantir une couverture complète des tests et une qualité logicielle élevée.

CONCLUSION

En récapitulant les principales idées, le test de boîte noire est une méthode essentielle dans le processus d'assurance qualité des logiciels. En se concentrant sur le comportement externe du logiciel sans se soucier de sa structure interne, les techniques de test de boîte noire permettent de détecter des erreurs potentielles, d'assurer la conformité aux spécifications et de garantir la fiabilité et la fonctionnalité du logiciel.

L'importance du test de boîte noire dans l'assurance qualité des logiciels réside dans sa capacité à simuler les perspectives de l'utilisateur final, à identifier les erreurs de conception et à fournir une couverture exhaustive des cas d'utilisation. En combinant différentes techniques telles que les tests d'équivalence, les tests de valeurs limites, les tests de combinaison et les tests de scénarios, les testeurs peuvent garantir une qualité logicielle élevée et une expérience utilisateur satisfaisante.

En continuant à investir dans la recherche et le développement de nouvelles techniques et outils de test de boîte noire, les professionnels de l'assurance qualité peuvent contribuer à améliorer la qualité et la fiabilité des logiciels, ce qui est essentiel dans un monde de plus en plus dépendant de la technologie.