INFOMANIA

QUALITÉ LOGICIELLE

L'importance de la stratégie de test dans la réussite et la pérennité de votre projet

INTRODUCTION

Les projets informatiques deviennent de plus en plus complexes avec l'utilisation massive de ces solutions et la complexité de l'environnement dans lesquelles elles vont être déployées. C'est pourquoi, plus que jamais les démarches tests logicielles sont devenues essentielles tout au long du cycle de vie de développement d'une solution et représentent ainsi l'un des principaux leviers de la réussite d'un projet de développement d'applications.

Pour être efficace, les professionnels s'accordent pour dire qu'un processus de test fonctionnel et technique doit être réalisé en parallèle du processus de développement d'une application, et non pas seulement en fin du projet comme on pourrait le penser de prime abord. Une logique qui est également mise en avant par les méthodes agiles, qui prône une écoute du client et des tests tout au long du cycle de développement.

Néanmoins il faut être convaincu qu'un logiciel « zéro défaut » n'existe pas. La présence de fautes logicielles ou de bugs est tout à fait normal lors de la réalisation du projet, néanmoins il faut qu'elles soient considérées avec beaucoup d'attention dans le but de les rectifier, en particulier lorsque les conséquences de ces fautes peuvent influer sur la sécurité ou la fonctionnalité de la solution. A ce niveau La vérification et les tests sont essentielle. Ces techniques, si elles sont adoptées et appliquées rigoureusement et avec intelligence dès le début d'un projet, elles peuvent améliorer significativement la qualité logicielle et la fonctionnalité de la solution.

QUALITÉ LOGICIELLE, C'EST QUOI AU JUSTE

La définition de la qualité logicielle est relative à chacun selon son niveau d'utilisation de la solution, un utilisateur final jugera plus au niveau de l'ergonomie, la facilité d'utilisation pour accomplir certaines tâches, la stabilité, la fiabilité, les performances. Par contre, un développeur appréciera la rapidité de développement, la qualité et la compréhensibilité du code source, la pertinence de la conception, la facilité de maintenance et la testabilité. Un exploitant s'attachera davantage à la facilité d'installation et de mise à jour, à la simplicité du diagnostic et de la supervision. L'architecte sera plus sensible à la compatibilité de la solution avec le parc applicatif existant, à la portabilité. Un DSI, enfin, estimera les coûts de développement puis de maintenance et d'évolution.

C'est pourquoi il est important de définir un ensemble de caractéristiques qui permettront de juger de manière objective la solution. La norme ISO/CEI 9126 définit justement l'ensemble de ses paramètres et standardise la notion de la qualité logicielle.

Selon cette norme, les 6 grandes caractéristiques de la qualité logicielle sont les suivantes :

Capacité fonctionnelle : Le logiciel répond-il aux besoins de ses utilisateurs ?

• Fiabilité : Le logiciel est-il en mesure d'assurer un niveau de qualité de service suffisant pour satisfaire les besoins opérationnels de ses utilisateurs ?

• Facilité d'utilisation : Le logiciel peut-il être utilisé à moindre effort ?
• Maintenabilité : Est-il facile d'adapter le logiciel à de nouveaux besoins ou à de nouvelles contraintes ?
• Rendement / Scalabilité : Les ressources matérielles nécessaires à l'exécution du logiciel sont-elles en rapport avec sa rentabilité ?
• Portabilité : Le logiciel peut-il être transféré facilement d'une plate-forme ou d'un environnement à un autre ?

Comment construire les tests d'un logiciel

La démarche test du logiciel a pour but de démontrer que les applications issues d'une phase du cycle de développement sont conformes aux spécifications (incluant les exigences légales et réglementaires) établies lors des phases précédentes. Elle a également pour but de détecter et de rendre compte des fautes qui peuvent avoir été introduites au cours des phases précédant la vérification.

Dans ce qui suit nous essaierons de détailler les points importants à ne pas négliger pour réussir votre stratégie de test.

1. Exigences à tester

La liste suivante identifie les items, cas d'utilisation, exigences fonctionnelles et exigences non-fonctionnelles, qui ont été désignés comme cibles de test. Cette liste représente ce qui sera testé.

1.1 Objectif du test

• Identifier les informations existantes du projet et les composants qui doivent être testés.
• Énumérer les exigences d'évaluation à haut niveau
• Recommander et décrire les stratégies de test à employer
• Identifier les ressources nécessaires et fournir un estimé de l'effort de test
• dentifier les biens livrables pour les tests.

1.2 Portée du test

- Énumérer brièvement les caractéristiques ciblées pour test et les fonctions qui ne seront pas testées.
- Énumérer toutes les hypothèses faites durant le développement de ce document qui ont un impact sur la conception, le développement ou l'implémentation des tests.
- Énumérer les risques et contingences qui peuvent affecter la conception, le développement ou l'implémentation des tests.
- Énumérer toutes les contraintes qui peuvent affecter la conception, le développement ou l'implémentation des tests.

2. Stratégie de tests

2.1.1 Tests fonctionnels

[Les tests fonctionnels portent sur les exigences fonctionnelles qui peuvent être retracées dans les cas d'utilisation ou les fonctions d'affaire et dans les règles d'affaire. Le but de ces tests est de vérifier la validité des données, de leur traitement et de leur récupération ainsi que des règles d'affaire. Ce type d'évaluation repose sur la technique de la boîte noire, c'est-à-dire, une vérification de l'application et de ses processus internes en l'utilisant avec l'interface utilisateur et en en analysant les résultats. Voici un exemple de stratégie proposée :]

2.1.2 Tests d'interface utilisateur

[Les tests d'interfaces utilisateur évaluent l'interaction de l'utilisateur avec le logiciel. Le but de ces tests est de vérifier que l'interface utilisateur donne l'accès approprié aux fonctions des cibles de test. De plus, ces tests permettent de vérifier que les objets de l'interface utilisateur se comporte tel qu'attendu et qu'ils sont conformes aux normes du client ou de l'industrie.]

2.1.3 Tests de données et d'intégrité de base de données

[Les bases de données et les processus de base de données doivent être testés comme sous-systèmes à l'intérieur du projet. Ces sous-systèmes doivent être testés sans utiliser l'interface utilisateur comme interface des données. L'identification des outils et techniques pour effectuer les tests repose sur la nature du système de gestion de base de données (SGBD).]

2.1.4 Profilage de performance

[Le profilage de la performance est un test de performance où les temps de réponses, les taux de transaction et les autres exigences temporelles sont mesurées et évaluées. Le but du profilage de performance est de vérifier si les exigences de performances ont été atteintes. Il est implémenté et exécuté profiler et ajuster la performance des cibles de test en fonction de conditions comme la charge de travail et les configurations matérielles.

2.1.5 Tests de charge

[Les tests de charges sont des tests de performance où les cibles de test sont soumis à différentes charges de travail afin d'évaluer leur performance et de s'assurer que leur fonction s'exécute normalement malgré les variations de charge. Le but des tests de charge est de déterminer et de s'assurer que le système fonctionne normalement par delà la charge de travail maximale. De plus, les tests de charge évaluent les caractéristiques de performance comme les temps de réponse, les taux de transaction et autres comportements pertinents.
NOTE : Les transactions sont, ci-dessous, des transaction d'affaire logiques. « Ces transactions sont définies comme des cas d'utilisation qu'un acteur du système est présumé exécuter en utilisant une cible de test comme, par exemple, ajouter ou modifier un contrat.]

2.1.6 Tests de stress

[Les tests de stress sont un type de test de performance qui sont implémentés et exécutés pour trouver des erreurs imputables à un manque de ressources. Une anomalie de la cible de test peut être identifiée à cause d'une mémoire vive ou d'un espace disque insuffisants. D'autres anomalies peuvent être imputables à une concurrence d'accès à des ressources comme, par exemple, une largeur de bande réseau ou un accès concurrent à une base de données. Les stress tests peuvent aussi être utilisés pour identifier la pointe qu'une charge de travail peut atteindre.]

2.1.7 Tests de volumétrie

[Les tests de volumétrie soumettent les cibles de test à d'important volume de données afin de déterminer quelles sont les limites du logiciel avant qu'il y ait défaillance. Les tests de volumétrie permettent aussi d'identifier la charge maximum ou le volume que les cibles de tests peuvent supporter pour une période de temps donnée. Ainsi, par exemple, pour la publication d'un rapport qui nécessite le traitement d'enregistrement de la base de données, le test de volumétrie devrait utiliser une base de données de grande taille et devra vérifier si le logiciel se comporte normalement et produit correctement le rapport.]

2.1.8 Tests de sécurité et de contrôle d'accès

[Les tests de sécurité et de contrôle d'accès portent sur deux domaines clés de la sécurité :
1. La sécurité au niveau de l'application incluant l'accès aux fonctions de traitement de données.
2. La sécurité au niveau du système, incluant la connexion à distance au système.
La sécurité au niveau de l'application contrôle l'accès des acteurs aux fonctions ou cas d'utilisation ou données qui leur sont rendues disponibles. Par exemple, tous les acteurs peuvent saisir des données et créer de nouveaux comptes, mais seuls les administrateurs peut les détruire. La sécurité au niveau des données est testée en s'assurant qu'un type d'utilisateur peut voir toutes les informations d'un client, incluant les données financières, alors qu'un autre type ne peut voir que les données démographiques pour le même client..
La sécurité au niveau du système permet aux seuls acteurs dont l'accès est autorisé, d'accéder aux applications par les seules passerelles appropriées.]

2.1.9 Tests de basculement et de récupération

[Les tests de basculement et de récupération vérifient que la cible de test peut réussir à basculer et à récupérer pour différents dysfonctionnements matériels, logiciels ou de réseau sans qu'il y ait perte de données ou corruption de sonnées.
Les tests de basculement vérifient que, pour les systèmes qui doivent demeurer en opération, lorsqu'une défaillance survient, un système alternatif ou un système de sauvegarde prend le relais correctement sans perte de données ou de transactions.
Les tests de récupération est un test où le système est exposé à des conditions extrêmes, ou des conditions simulées pour provoquer une défaillance, telle qu'une panne d'un dispositif d'entrée-sortie ou des pointeurs de base de données invalides Les processus de récupération sont alors invoqués et l'application ou le système est alors suivi ou inspecté pour vérifier si la récupération des données, de l'application ou du système a été bien effectuée.]

2.1.10 Tests de configuration

[Les tests de configuration valident les opérations pour des cibles de test pour différentes configurations logicielles et matérielles. Les spécifications matérielles des postes de travail, des réseaux et des serveurs de base de données, en environnement de production, varient d'un client à l'autre. Les postes de travail peuvent avoir différents logiciels en opération selon différentes combinaisons et utilisant différentes ressources.]

2.1.11 Tests d'installation

[Les tests d'installation poursuivent deux objectifs. D'abord, s'assurer que le logiciel peut être installé dans différentes conditions, tant pour les nouvelles installations, les mises à jour que pour compléter une installation personnalisée dans des conditions normales et des conditions anormales. Les conditions anormales comprennent un espace disque insuffisant, des autorisations insuffisantes pour créer des répertoires, etc. L'autre objectif est de vérifier que, une fois installé, le logiciel fonctionne correctement. Cela signifie qu'il faut exécuter un certain nombre de tests fonctionnels.]

2.2 Outils

Les outils suivants seront employés pour ce projet:

3. Biens livrables

3.1 Modèle de test

[Identifier les rapports qui seront créés et extraits du modèle de tests.]

3.2 Journaux de test

[Décrire la méthode et les outils utilisés pour enregistrer et faire rapport des résultats de test et du statut des tests.]

3.3 Rapports d'anomalies

[Identifier la méthode et les outils utilisés pour enregistrer, tracer et faire rapport des incidents de test et de leur statut.]

3. Annexe A: Tâches du projet

Voici la liste des tâches liées à l'exécution des tests :

• Planifier les tests
- o Identifier les exigences de test
- o Évaluer les risques
- o Identifier les ressources nécessaires aux tests
- o Développer une stratégie de test
- o Créer le calendrier de tests
- o Générer le plan de test

• Concevoir les tests
- o Analyser la charge de travail
- o Identifier et décrire les cas de test
- o Identifier et structurer les procédures de test
- o Réviser la couverture des tests

• Implémenter les tests
- o Enregistrer ou programmer les scripts de test
- o Identifier les fonctionnalités propres au test dans les modèles de conception et d'implémentation.
- o Établir des ensembles de données externes

• Exécuter les tests
- o Exécuter les procédures de test
- o Évaluer l'exécution des tests
- o Récupérer les tests suspendus
- o Vérifier les résultats
- o Examiner les résultats non prévus
- o Enregistrer les anomalies

• Évaluer les tests
- o Évaluer la couverture des cas de tests
- o Évaluer la couverture du code
- o Analyser les anomalies
- o Déterminer si les critères de complétion et de succès des tests ont été atteints.

INFOMANIA

66 RUE SAINT GEORGES
69005 LYON
tel :04 22 14 07 27
fax:04 65 02 00 25
contact@infomania-services.fr
www.infomania-services.fr