SOFTWARE TESTING TECHNIQUES

***Violina ELADII,***

***Maria MACOVEI,***

*Université technique de Moldavie,*

*Faculté: Informatique, Informatique, Microélectronique,*

*Département: Génie logiciel et automatisation,*

*Groupe: FI-181,*

*Ville: Chisinau, République de Moldova.*

Eladii Violina,

violinaeladii@gmail.com

*Résumé: Avec la complexité croissante de l’injonction des applications logicielles d’aujourd’hui, la pression concurrentielle croissante a poussé l’assurance qualité des logiciels développés vers de nouveaux sommets. Les tests logiciels sont une partie inévitable du cycle de vie du développement logiciel, et rester en phase avec sa criticité dans le processus de pré et post-développement en fait quelque chose qui devrait être adapté avec des méthodologies et des techniques améliorées et efficaces. Ce document vise à discuter des techniques de test existante ainsi que des techniques de test améliorées à des fins de meilleure assurance qualité.*

***Mots clés: Méthodologies de test, cycle de vie des tests logiciels, cadres de test, tests d'automatisation, développement piloté par les tests, optimisation des tests, mesures de qualité.***

INTRODUCTION

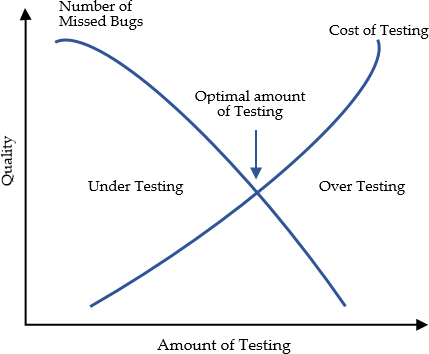
Le test est défini comme un processus d'évaluation du fait que le système spécifique répond ou non aux exigences initialement spécifiées. Il s'agit principalement d'un processus englobant le processus de validation et de vérification pour déterminer si le système développé répond aux exigences définies par l'utilisateur. Par conséquent, cette activité entraîne une différence entre le résultat réel et le résultat attendu. Le test de logiciel fait référence à la recherche de bogues, d'erreurs ou d'exigences manquantes dans le système ou le logiciel développé. Il s'agit donc d'une enquête qui fournit aux parties prenantes les connaissances exactes sur la qualité du produit.

Les tests de logiciels peuvent également être considérés comme une activité basée sur les risques. La chose importante pendant le processus de test, les testeurs de logiciels doivent comprendre comment minimiser un grand nombre de tests dans un ensemble de tests gérables, et prendre des décisions judicieuses sur les risques qu'il est important de tester ou ce qui ne l'est pas.

La figure 1 montre le coût des tests et les erreurs ont trouvé une relation. La figure 1 montre clairement que le coût augmente considérablement dans les tests desdeux types, c'est-à-dire fonctionnels et non fonctionnels. La prise de décision sur ce qu'il faut tester ou réduire les tests peut alors faire rater de nombreux bogues. Le but des tests efficaces est de faire cette quantité optimale de tests afin que l'effort de test supplémentaire puisse être minimisé.

Selon la figure 1, les tests logiciels sont un élément important de l'assurance qualité des logiciels. L'importance des tests peut être prise en compte à partir des tests logiciels vitaux (par exemple, les commandes de vol) qui peuvent être très coûteux en raison du risque lié aux retards d'horaire, aux dépassements de coûts ou à l'annulation pure et simple , et plus à ce sujet.

Les tests comportent certains niveaux et étapes selon lesquels la personne qui effectue les tests diffère d'un niveau à l'autre. Les trois étapes de base des tests logiciels sont les tests unitaires, les tests d'intégration et les tests système. Chacune de ces étapes est testée soit par le développeur du logiciel, soit par l'ingénieur d'assurance qualité, également connu sous le nom de testeur de logiciel. Les étapes de test mentionnées ci-dessus sont incluses dans le cycle de vie du développement logiciel (SDLC). Il est essentiel de diviser le développement logiciel en un ensemble de modules où chaque module est attribué à une équipe différente ou à une personne différente. Après l'achèvement de chaque module ou unité, il est testé par le développeur juste pour vérifier si le module développé fonctionne selon les attentes ou non, cela s'appelle un test unitaire. La deuxième étape des tests dans le SDLC est le test d'intégration. Une fois que les modules d'un même système logiciel ont été développés indépendamment, ils sont intégrés ensemble et souvent des erreurs surviennent dans la construction une fois l'intégration terminée. La dernière étape de test du SDLC est le test du système, qui teste l'ensemble du logiciel sous tous les angles. De plus, les tests logiciels garantissent que les unités intégrées n'interfèrent ni ne perturbent la programmation d'aucun autre module. Cependant, le test d'un système volumineux ou extrêmement complexe peut être une procédure extrêmement longue et longue, car plus il y a de composants dans l'application, plus il devient difficile de tester chaque combinaison.



**Figure 1:** Chaque projet logiciel a un effort de test optimal (avec l'aimable autorisation ).

et le scénario, conduisant par conséquent à un besoin urgent d'un processus de test logiciel amélioré pour l'optimisation premium. Le cycle de test est principalement composé de plusieurs phases, de la planification des tests à l'analyse des résultats des tests. La planification des tests étant la première phase est principalement le plan de toutes les activités de test qui doivent être menées dans l'ensemble du processus de test. Le développement de tests est la deuxième phase du cycle de vie des tests, au cours de laquelle les cas de test qui doivent être utilisés dans le processus de test sont développés. L'exécution du test est la phase suivante du cycle de test qui englobe l'exécution des cas de test, et les bogues pertinents sont signalés dans la phase suivante qui est la phase de rapport de test. L'analyse des résultats de test est la dernière étape du processus de test dans laquelle l'analyse des défauts est effectuée par le développeur qui a développé le système ou le logiciel, cette étape peut également être gérée avec le client car elle aidera à mieux comprendre ce qu'il faut

ignorer et quoi corriger, améliorer ou simplement modifier .

# 

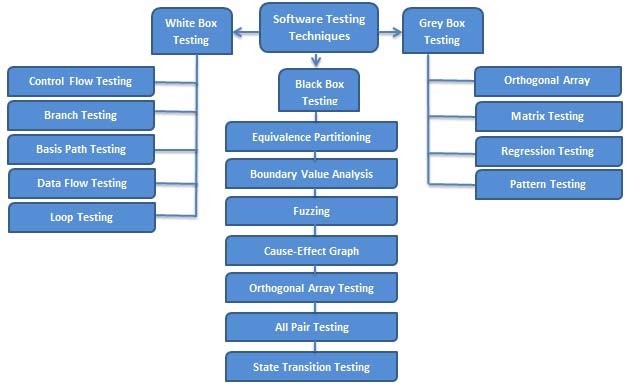
I.MÉTHODES D'ESSAI EXISTANTES

Pour le début du processus de test, la première étape consiste à générer des cas de test. Les cas de test sont développés en utilisant diverses techniques de test, pour des tests efficaces et précis. Les principales techniques de test sont les tests en boîte noire, en boîte blanche et en boîte grise.

Les tests White Box sont très efficaces car c'est la méthode de test qui teste non seulement la fonctionnalité du logiciel, mais également la structure interne de l'application. Lors de la conception des cas de test pour effectuer des tests en boîte blanche, des compétences en programmation sont nécessaires pour concevoir les cas de test. Le test de la boîte blanche est également appelé test de la boîte transparente ou de la boîte en verre. Ce type de test peut être appliqué à tous les niveaux, y compris les tests unitaires, d'intégration ou de système. Ce type de test est également appelé test de sécurité, car il répond au besoin de déterminer si les systèmes d'information protègent les données et maintiennent la fonctionnalité prévue. Comme ce type de processus de test utilise la disposition logique interne du logiciel, il est donc suffisamment capable de tester tous les chemins indépendants d'un module, chaque décision logique est exercée, toutes les boucles sont vérifiées à chaque niveau de frontière et les structures de données internes sont également exercés. Cependant, les tests en boîte blanche servent à être un processus de test complexe en raison de l'inclusion de compétences en programmation dans le processus de test.

Le test Black Box est une technique de test qui teste essentiellement le

fonctionnalité de l'application sans entrer dans le détail de son niveau d'implémentation. Cette technique peut être appliquée à tous les niveaux de test au sein du SDLC. Il exécute principalement les tests de manière à couvrir toutes les fonctionnalités de l'application pour déterminer si elle répond ou non aux exigences initialement spécifiées de l'utilisateur. Il est capable de trouver des fonctionnalités incorrectes en testant leur fonctionnalité à chaque valeur minimale, maximale et de base. Il s'agit du processus de test le plus simple et le plus répandu dans le monde.

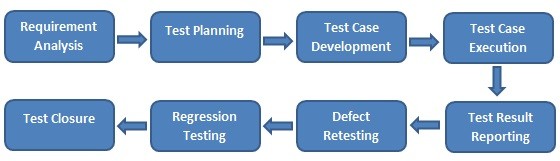


**Figure 2**: Techniques de test de logiciels.

Le test de la boîte grise est la combinaison de la technique de test de la boîte blanche et de la boîte noire au service des avantages des deux. Le besoin d'un tel type de test a été suscité car dans ce type de test, le testeur est conscient de la structure interne de l'application, testant ainsi la fonctionnalité d'une meilleure manière en tenant compte de la structure interne de l'application. La figure 2 est référencée par l'auteur J. Irena [8] et étendue ici dans notre article de recherche.

A. Cycle de vie des tests logiciels (STLC)

La figure 3 décrit les étapes, les étapes et les phases STLC qu'un logiciel subit pendant le processus de test. Cependant, il n'y a pas de norme fixe du logiciel ou de l'application soumis au STLC, et cela varie d'une région à l'autre à travers le monde [11].



**Figure 3:** Cycle de vie des tests logiciels.

Au cours de la première phase du STLC, l'examen des exigences logicielles est effectué par l'équipe d'assurance qualité dans laquelle ils comprennent les exigences de base en fonction desquelles le test sera effectué. En cas de conflit, l'équipe doit se coordonner avec l'équipe de développement pour mieux comprendre et résoudre le conflit. La planification des tests est la deuxième et la plus importante phase du STLC, car c'est l'étape où toute la stratégie de test est définie. Cette phase traite de la préparation du plan de test, qui sera l'ultime livrable de cette phase. Le plan de test est un document obligatoire orienté vers le test fonctionnel de l'application, sans lequel le processus de test n'est pas possible .

La phase de conception de test est la phase où le cas de test est développé et l'activité de planification de test est interrompue. Les cas de test appropriés sont rédigés manuellement par l'équipe d'assurance qualité ou, dans certains cas, des cas de test automatisés sont générés. Le scénario de test spécifie un ensemble d'entrées ou de données de test, de conditions d'exécution et de résultats attendus. L'ensemble spécifié de données de test doit être choisi de telle sorte qu'il produise le résultat attendu ainsi que des données intentionnellement erronées qui produiront une erreur pendant le test. Ceci est généralement fait pour vérifier les conditions dans lesquelles l'application cesse d'exécuter.

La phase d'exécution des tests comprend l'exécution des cas de test en fonction du plan de test qui a été produit avant la phase d'exécution. Si la fonctionnalité passe la phase d'exécution sans aucun rapport de bogue, le test est dit effacé ou réussi, et chaque cas de test échoué sera associé au bogue ou à l'erreur trouvé. Le livrable d'une telle activité est un rapport de défaut ou de bogue.

Le rapport de test est le rapport des résultats générés après l'exécution des cas de test qui implique également le rapport de bogue qui est ensuite transmis à l'équipe de développement afin qu'il puisse être corrigé .

A. Cycle de vie de la version du logiciel

Ce cycle de vie émerge après le STLC et englobe un processus de test supplémentaire dans lequel les tests alpha et bêta sont inclusifs.

Alpha Testing, dans lequel Alpha fait référence au test de première étape de l'application chez le développeur, peut être effectué via la technique de la boîte blanche ou la technique de la boîte grise. Les tests au niveau de l'intégration ou des tests au niveau du système pourraient être effectués en utilisant l'approche de la boîte noire, appelée version alpha. Le test alpha s'arrête avec un gel des fonctionnalités, ce qui signifie généralement qu'aucune fonctionnalité supplémentaire ne sera ajoutée pour étendre la fonctionnalité ou à toute autre fin.

La phase de test bêta vient après le test Alpha et peut être considérée comme un test d'acceptation formel tel qu'il est effectué par l'utilisateur, après la version Alpha. Le logiciel ou l'application est distribué à un certain groupe d'utilisateurs prévu à des fins de test. Habituellement, la version bêta des applications est mise à la disposition du public ciblé pour commentaires avant sa sortie officielle. Le public ciblé est souvent appelé Beta Testers, et l'application peut être qualifiée de version prototype du logiciel principalement à des fins de démonstration. Par conséquent, la version finale du logiciel est publiée après le test bêta.

I. AMÉLIORATION DES PROCESSUS D'ESSAI

La priorisation de la suite de tests améliore le processus de test par des critères combinatoires. La méthodologie principale derrière une telle hiérarchisation des cas de test est la conversion des blogs en suites de tests pertinentes avec la session utilisateur, et leur écriture supplémentaire dans un format XML. L'algorithme utilisé pour cette approche doit être hiérarchisé avec précision par la couverture basée sur des suites de tests combinatoires. En outre, des études empiriques devraient être menées pour analyser l'efficacité de l'application spécifique et de ses suites de tests pertinentes. Un outil utilisé à cet égard est connu sous le nom de C-PUT qui formate essentiellement les journaux des applications Web en suites de tests qui sont formatées en XML; il est ensuite utilisé pour la fourniture de la fonctionnalité de hiérarchisation de ces tests. Des recherches sont en cours pour savoir si ces techniques de hiérarchisation des suites de tests peuvent être utilisées pour améliorer le taux de détection des défauts ou non.L'utilisation d'algorithmes génétiques (GA) aux fins de la génération automatisée de données de test pour tester l'application est une autre amélioration du processus de test, car auparavant les moyens dynamiques de génération de données de test restaient un gros problème dans le processus de test logiciel, de sorte que l'utilisation de tests basés sur un algorithme génétique est un outil efficace de la génération de données de test, il est également capable de gérer la génération de données en fonction de la complexité du programme.

A. Automatisation des tests

L'amélioration majeure du processus de test conduit le processus de test vers l'automatisation des tests, qui consiste à utiliser un logiciel particulier pour effectuer le processus de test et à comparer les résultats réels avec les résultats attendus. La technique d'automatisation des tests est efficace dans le temps, car elle permet d'économiser le temps des tests manuels qui peuvent être assez laborieux.

Dans SDLC, l'automatisation des tests se produit pendant la mise en œuvre ainsi que la phase de test. Partout dans le monde, l'automatisation des tests est pratiquée au lieu des tests manuels, car elle permet de gagner beaucoup de temps en accomplissant les processus de test dans un laps de temps plus court. L'automatisation des tests a repris le processus de test manuel en réduisant ses besoins ainsi qu'en exposant le nombre d'erreurs, des lacunes qui ne peuvent pas être reconnues via le processus de test manuel.

Les tests de régression étant l'un des principaux types de tests, ils nécessitent beaucoup de temps lorsqu'ils sont effectués manuellement. Il teste généralement si le logiciel ou l'application fonctionne correctement après la correction de bogues ou d'erreurs. Parce que parfois après la correction de l'erreur, le taux d'erreur ou de bogue du code ou de l'application est encore plus élevé. Donc, pour éviter le temps nécessaire pour les tests de régression; un ensemble de suites de tests automatisés est créé pour former une suite de tests de régression à cette fin. L'automatisation des tests aide également à trouver le problème à un stade beaucoup plus précoce, économisant des tas de coûts de modification et d'énergie aux étapes ultérieures.

L'environnement qui correspond à un terme connaît généralement l'exécution des tests d'automatisation appelée Testing Framework. Le cadre de test est principalement responsable de l'exécution des tests, ainsi que de la définition du format dans lequel exprimer les attentes et du reporting des résultats. La caractéristique remarquable de Testing Framework qui le rend largement applicable dans divers domaines à travers le monde est son indépendance des applications. Les frameworks de test sont de certains types, notamment modulaires, pilotés par les données, pilotés par mots-clés et hybrides. Le cadre de test modulaire est basé sur le principe de l'abstraction qui implique la création de différents scripts pour différents modules du logiciel ou de l'application à tester, faisant ainsi abstraction de chaque composant d'un autre niveau. Cette division modulaire conduit à l'évolutivité ainsi qu'à une maintenance plus facile des suites de tests automatisées. De plus, une fois que la fonctionnalité est disponible dans la bibliothèque, la création de différents scripts de pilote pour différents types de tests devient facile et rapide. L'inconvénient majeur d'un tel type de framework est d'incorporer des données à l'intérieur, donc lorsque la modification ou la gradation est requise dans les données de test, tout le code du script de test doit être modifié. C'était la cause principale qui a servi de but à l'invention du Data Driven Testing Framework. Dans ce type de framework, les données de test et les résultats attendus sont idéalement stockés dans différents fichiers, aidant à l'exécution d'un script de pilote unique pouvant exécuter tous les cas de test avec plusieurs ensembles de données. Ce type de Framework réduit le nombre de scripts de test ainsi que minimise la quantité de code souhaitée pour la génération de cas de test, donne plus de flexibilité dans la fixation des erreurs ou des bogues.

Le cadre de test piloté par mots-clés utilise des

mots-clés appelés directives. Ce type de cadre est utilisé pour expliquer les actions qui devraient être exécutées par le logiciel ou l'application à tester. Ce type de test est essentiellement une extension de Data Driven Testing car les données ainsi que les directives sont conservées dans des fichiers de données séparés. Il englobe tous les avantages du cadre de test piloté par les données. En outre, la réutilisabilité des mots-clés est un autre avantage majeur. Le mauvais facteur de ce type de cadre de test est qu'en raison de l'utilisation de mots-clés, il ajoute de la complexité au cadre, ce qui rend les cas de test plus longs et plus complexes. Par conséquent, pour combiner les forces de tous les cadres d'atténuation des facteurs néfastes qu'ils possèdent. Une approche hybride est considérée comme la meilleure pour l'utilisation car elle est principalement une combinaison des trois approches et cette combinaison intègre les avantages de tous les cadres de test, ce qui en fait la plus efficace.

A. Tester les frameworks dans Agile

Le cycle de vie agile est une autre innovation dans le domaine des tests logiciels car il englobe des cycles de test courts et rapides avec des exigences fréquemment modifiées. Ainsi, l'environnement agile peut englober n'importe quel cadre de test, mais en raison de la fréquence

itérations et changement rapide des exigences spécifiées, la maintenance de la suite d'automatisation de tests devient assez difficile. Bien que les frameworks de test restent mal adaptés à l'environnement agile, il reste difficile d'atteindre une couverture maximale du code et des fonctionnalités.

Développement piloté par les tests (TDD)

C'est une technique qui utilise des tests unitaires automatisés dans le but de piloter la conception du logiciel et de forcer le processus de découplage des dépendances. Avec le processus de test habituel, le testeur trouve souvent un ou plusieurs défauts ou erreurs, mais TDD donne une mesure limpide du succès lorsque le test n'échoue plus, améliorant le niveau de confiance quant au système répondant à ses spécifications de base. En utilisant l'approche TDD, une grande quantité de temps peut être économisée, ce qui pourrait être gaspillé au cours du processus de débogage.

BDD (Behavior Driven Development) est principalement une extension du Test-driven Development qui se concentre sur les aspects comportementaux du système plutôt que sur les aspects au niveau de la mise en œuvre. Par conséquent, donner une compréhension claire de ce que le système est censé faire exactement, ce qui donne plus d'efficacité au processus de test. Ainsi, BDD est principalement un développement piloté par les tests incorporé aux tests d'acceptation, qui se réfère généralement à la réalisation d'un test pour déterminer si l'exigence spécifiée du produit ou du logiciel est satisfaite ou non. S'il est effectué par le client ou l'utilisateur prévu, alors il est appelé test d'acceptation par l'utilisateur.

MESURES DE TEST

A. Mesures de priorisation

L'utilisation des mesures de test a une importance primordiale car elles peuvent considérablement améliorer l'efficacité du processus de test. Ils servent d'indicateur important de l'efficacité et de l'exactitude et de l'analyse des paramètres définis. Ils peuvent également aider à identifier les domaines qui nécessitent une amélioration ainsi que les actions ou mesures ultérieures qui doivent être prises pour les éliminer. Les mesures de test ne sont pas seulement une étape unique dans STLC, mais agissent comme un parapluie pour l'amélioration constante de l'ensemble du processus de test lui-même. Les mesures de la qualité des processus et les mesures de la qualité des produits visent toutes deux à fournir des améliorations non seulement dans le processus de test, mais également dans la qualité du produit.

Cependant, il y a un problème critique auquel sont confrontés les

processus de test qui correspond à l'approche de test avec l'application en cours de développement. Toutes les approches de test ne peuvent pas être implémentées dans chaque application à développer. Par exemple, le test d'un logiciel de protocole réseau par rapport au test de certaines applications de commerce électronique sera assez différent avec une complexité de cas de test complètement différente, et

qui souligne la criticité de l'implication humaine dans le processus de test et pas seulement la simple dépendance aux cas de test existants. La hiérarchisation basée sur la fréquence améliore le processus de test de telle sorte que les cas de test qui englobent les pages les plus utilisées sont sélectionnés pour être exécutés avant les cas de test qui utilisent les moins fréquents .

A. Mesures de la qualité des processus

Un processus est la partie la plus éminente car il est capable de produire un résultat de qualité dans le moins de temps et de la manière la plus rentable. C'est la raison ultime pour laquelle les organisations du monde entier se sont concentrées sur l'amélioration de la performance des processus, et c'est exactement là où le besoin de mesures est apparu, car il est nécessaire d'évaluer efficacement le processus à partir de différentes dimensions. Mesure de l'efficacité du processus est la métrique clé de la qualité du processus qui englobe certaines mesures de facteurs comme la courbe de progression du test qui décrit la progression prévue de la phase de test par le plan de test.

Le coût des tests est la prochaine étape majeure de la métrique à la fois par phase et par composant. L'objectif majeur est d'aider à identifier les pièces qui nécessitent des tests intensifs et le coût qu'elles supporteront en fonction de celui-ci. Le temps de traitement moyen des défauts est une autre mesure qui représente le temps moyen de vérification par l'équipe de test pour la vérification des défauts. Le temps moyen de réponse aux défauts est la métrique qui est un indicateur de l'efficacité opérationnelle. C'est la mesure du temps moyen mis par l'équipe pour répondre aux erreurs. Les mesures de l'efficacité des processus garantissent que le résultat

l'application ou les produits donneront une sortie de haute qualité. Test la couverture, l'efficacité d'élimination des défauts, l'indice de volatilité des exigences, les cas de test échoués et exécutés en sont les principales catégories assurant un processus de test global amélioré.

CONCLUSION ET TRAVAUX FUTURS

Les tests sont la partie la plus critique du cycle de vie du développement logiciel, car c'est quelque chose dont dépend la livraison finale du produit. Cela prend du temps et un processus intensif, par conséquent, des techniques améliorées et des méthodologies innovantes sont nécessaires. Cela rend les tests automatisés et d'autres diverses implémentations de mesures de test avant et pendant le processus de test. Il peut améliorer les méthodes de test existantes, à la fois pour l'efficacité du temps ainsi que pour le produit final efficace et fiable qui non seulement répond aux exigences spécifiées, mais offre également une efficacité opérationnelle maximale.

La plate-forme sur laquelle résident le développement et les tests de logiciels continue d'évoluer et reste extrêmement importante. Cependant, quelque chose d'aussi crucial et critique comme le test arrive souvent assez tard dans le processus de développement logiciel. Il doit y avoir une interaction maximale entre les rédacteurs de spécifications et les testeurs pour une meilleure compréhension et un examen précoce, ce qui peut résoudre les problèmes d'ambiguïté et, par conséquent, réduire les coûts de réparation ultérieure du logiciel. Les testeurs, après avoir été clairs sur les spécifications et les exigences, doivent remettre aux développeurs un certain modèle de test léger, afin qu'ils s'assurent que les spécifications principales sont respectées avant de traiter le projet pour les tests officiels. L'utilisation d'outils de simulation peut énormément aider les testeurs à créer l'environnement similaire dans lequel le produit est destiné à fonctionner, certains tests et méthodes d'exception peuvent être mieux déterminés. Tout en testant le produit dans l'environnement de test similaire auquel le produit est destiné, et cela peut être facilement réalisé en intégrant la simulation dans le processus de test. Par conséquent, les travaux futurs en rapport avec le processus de test seront beaucoup plus dépendants de la technologie, exploitant l'approche basée sur la simulation et le modèle de test automatisé, non seulement pour accélérer le cycle de vie des tests, mais également pour une prévention optimale des bogues et une assurance qualité efficace.

**Références Web:**

<https://www.guru99.com/software-testing-introduction-importance.html>

<https://www.guru99.com/manual-testing.html>

<https://www.guru99.com/difference-automated-vs-manual-testing.html>

<https://www.guru99.com/webservice-testing-beginner-guide.html>

<https://www.guru99.com/introduction-to-soapui.html>

<https://blog.testlodge.com/what-is-manual-testing/>

<https://www.tutorialspoint.com/soapui/soapui_project.htm>