Les tests fonctionnels

Table des matières

Introduction

Les tests fonctionnels ont pour but de vérifier la conformité de l'application développée avec le cahier des charges et l'analyse formelle. Ils sont donc basés sur les spécifications fonctionnelles.

Les tests fonctionnels mettent en œuvre des scénarios qui permettent de tester les fonctions pour lesquelles le logiciel a été développé.

Les testeurs vérifient l'exécution de chacune de ces fonctions. L'utilisateur final en situation de travail, peut observer les problèmes qu'il rencontrerait, exprimer ses questions et demandes de changements du logiciel pour le rendre plus ergonomique (exemple : nombre de clics trop important – navigation trop longue jusqu'à un écran utilisé plusieurs fois par jour ; place des boutons 'ok' et 'annuler'…).

Les tests fonctionnels sont principalement développés à l'aide des techniques de test de la boîte noire, càd sans accès au code.

Niveaux

Chaque test fonctionnel valide une partie du logiciel :

- Soit une action.
    - La validation d'un champ (exemple : le nom du partenaire).
- Soit le scénario d'un cas d'utilisation.
    - L'exécution d'un cas d'utilisation (exemple : créer un partenaire).
- Soit un enchaînement de cas.
    - L'exécution d'un cas d'utilisation à l'intérieur d'un autre cas (exemple : créer un partenaire lors de la confirmation d'une mobilité).
    - Un enchainement de cas est également appelé un « scénario d'usage », càd une façon de travailler courante pour l'utilisateur final. Les scénarios d'usage permettent de tester l'application dans des conditions réelles ou quasi réelles, en adéquation avec les besoins-métiers.

Exemple :

A l'intérieur du cas d'utilisation « se connecter », après avoir testé que le scénario nominal se passe bien, il faudra tester l'introduction d'un login ou d'un mot de passe erroné.

Selon l'analyse fonctionnelle, si l'utilisateur rentre un login ou un mot de passe erroné, un message d'erreur est affiché et l'utilisateur n'est pas connecté à l'application.

Le test consistera donc à introduire un login erroné, à contrôler qu'un message d'erreur est bien affiché, qu'il s'agit du message d'erreur correspondant au login erroné et que l'utilisateur ne soit pas connecté à l'application.

Le test consistera ensuite à introduire un mot de passe erroné et à contrôler que le message d'erreur correspondant soit affiché et que l'utilisateur ne soit pas connecté à l'application.

Le test consistera enfin à introduire un mot de passe et un login erronés, à ne pas en introduire du tout etc.

Approche

Les tests devraient couvrir l'ensemble de toutes les combinaisons possibles d'emploi des fonctionnalités du logiciel (ordre de saisie, ordre d'apparition des erreurs, chemins…), mais ils le peuvent difficilement, c'est pourquoi il est intéressant d'observer les chemins pris par un utilisateur rodé au métier. On ciblera essentiellement cette façon de travailler et l'ensemble des cas-limites.

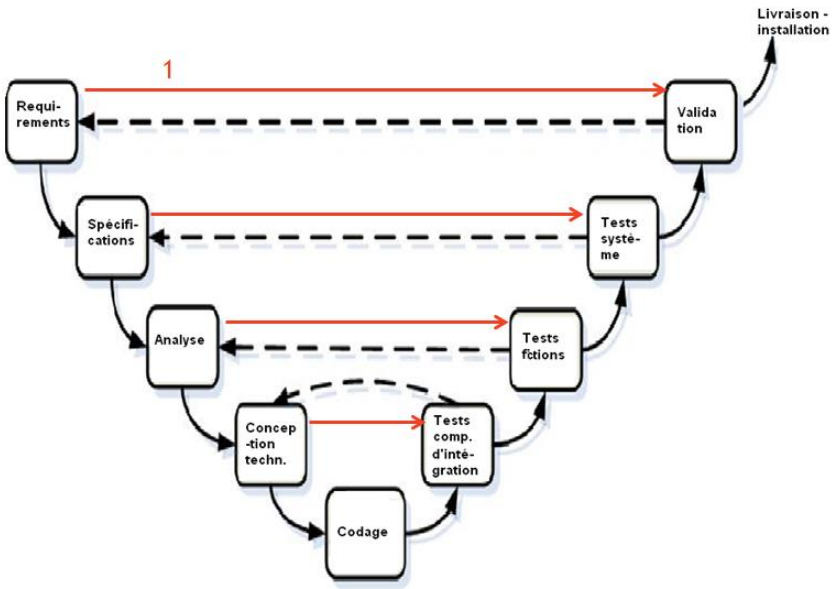
Il est possible d'automatiser une partie des tests fonctionnels mais il est important qu'une partie de ceux-ci soient réalisés manuellement pour tester l'utilisabilité et l'ergonomie du logiciel.

L'approche automatisée est souvent utilisée (notamment en Agile) pour relancer les tests fonctionnels à chaque build ou à chaque itération.

Prévoir les tests fonctionnels

Les équipes de développement et de tests pourraient être tentées d'écrire les tests au moment où ils sont effectués.

Cependant, comme nous le voyons pendant l'analyse du diagramme en V, il est préférable d'écrire les tests en regard des spécifications de chaque niveau du développement. Cela nous assurera que les tests correspondent aux demandes et non aux développements qui ont été faits. De cette manière, les tests pourront détecter les non-conformités du développement par rapport à ce qui avait été demandé.



(1) (couleur rouge) Toute description d'un composant est accompagnée de tests qui permettront de s'assurer qu'il correspond à sa description.

Faire évoluer les tests

Nous observerons l'importance du contrôle des changements apportés au logiciel, qu'il s'agisse de la correction de bugs ou de l'introduction de nouvelles demandes.

Les tests fonctionnels (comme tous les autres) devront donc évoluer en fonction des modifications apportées au logiciel.

Lorsqu'un bug est découvert et corrigé, il est essentiel de garder les données qui ont permis de le reproduire et le scénario de test. Ceux-ci seront intégrés dans le scénario global de tests de l'application. Ceci permettra de tester la non-régression dans les versions suivantes.

Préparer les tests

Environnement

Si les tests unitaires sont faits dans l'environnement de développement, les tests fonctionnels sont réalisés dans un environnement de tests indépendant.

Le logiciel doit donc être installé dans l'environnement de tests avant que ceux-ci ne soient exécutés. Il faut également définir des procédures d'installation de l'environnement de tests.

Si le logiciel tourne dans plusieurs environnements différents, tous doivent bénéficier des tests fonctionnels.

Exemple :

- Application tournant sous Windows et sous Linux.
- Tests de tous les browsers supportés.

Jeux d'essai

Il est essentiel de définir des jeux d'essai qui puissent être réutilisés à chaque exécution du test et donc de définir des procédures pour remettre l'environnement dans l'état initial avant chaque exécution.

Ces procédures sont essentielles, elles permettent de refaire exactement les mêmes tests dans les mêmes conditions.

L'environnement de test doit être reproductible autant de fois que nécessaire :

- Jusqu'au moment où tous les tests sont satisfaits pour une version.
- A chaque nouvelle version.

Cela permet de tester les non-régressions d'une version à l'autre ou d'un build à l'autre.

Les jeux d'essai évolueront également en fonction des nouvelles demandes et des corrections de bugs.

Ecrire les tests

Il faut prévoir que le document de tests puisse être rempli, commenté, daté et signé à chaque exécution.

Le document est donc la présentation des tests à réaliser, le remplissage du document se fait lors de chaque exécution des tests.

Il y aura plusieurs documents de test :

- Un par cas d'utilisation.
- Un par scénario d'usage.

Notons que la structure du document de test est applicable à tous les tests faits hors de l'environnement de développement :

- *Robustesse, performance, volume, vitesse de croisière…*

1. Description de l'environnement

Il faut d'abord prévoir une section dans laquelle le testeur pourra introduire la description de l'environnement de tests :

- Version du logiciel à tester
- Description du/des PC
- Description des périphériques
- OS
- Serveur de base de données
- Base de données
- …

2. Référence au(x) cas d'utilisation

Il faut explicitement faire le lien avec la fonctionnalité que l'on teste (exemple : numéro du cas d'utilisation ou code de la fonction) de sorte d'améliorer immédiatement tout test en lien avec d'éventuelles demandes de changement ou corrections de bugs.

3. Prérequis

Il faut documenter tous les prérequis à exécuter avant de pouvoir tester.

Exemple : (Projet AE – gestion de la journée d'entreprises) avant de pouvoir tester l'encodage d'une personne de contact, il faut avoir créé une entreprise.

4. Description des données en input

Il faut documenter les données déjà présentes dans l'application.

Exemple : (Projet AE– gestion de la journée d'entreprises)

- Nombre de journées d'entreprise (JE) :
- Nombre d'entreprises (E):
- Nombre d'utilisateurs (U):
- Nombre de participations (Pa) :

Il faut documenter les données qui seront introduites progressivement dans le test.

Exemple : (Projet AE– gestion de la journée d'entreprises)

- Décrire la/les entreprises qui seront créées pendant le test du cas d'utilisation « créer une entreprise ».

5. Description d'un scénario du test

Un scénario du test comprendra plusieurs étapes dépendant de la complexité du cas à tester.

Il faut tout d'abord prévoir le cas de test le plus général dans lequel tout se passe bien.

Il faut ensuite tester toutes les combinaisons dans lesquelles tout se passe encore bien.

Exemple : (Projet AE– gestion de la journée d'entreprises)

- Créer une entreprise avec dénomination, adresse, email et numéro de téléphone.
- Créer une entreprise avec dénomination, adresse et email.
- Créer une entreprise avec dénomination, adresse et numéro de téléphone.

Il faut ensuite tester les cas d'erreurs.

- Test des erreurs isolées.
  - Sur chaque champ d'un formulaire, tester la saisie de données erronées ou l'absence de données obligatoires.
  - Test des cas extrêmes :
    - Fourchette avec minimum plus grand que le maximum.
    - Valeurs autorisées, valeur autorisée maximale, valeur autorisée maximale + 1, valeur minimale -1.
    - Combinaisons inexistantes.
    - Valeurs nulles ou vides.
    - Chaîne de caractères plus grande que le nombre de caractères prévus.
    - Format de date invalide.
    - Date trop ancienne ou date dans le futur.
    - Nombre plus grand que celui supporté par le format du champ.
    - …

  - Par exemple sur chaque champ d'un formulaire, tester les erreurs possibles :
    Exemple : (Projet AE– gestion de la journée d'entreprises)
    dans le champ « nom de l'entreprise »
    - Ne pas introduire de valeur dans le champ « nom de l'entreprise ».
    - Introduire une valeur dépassant la limite du nombre de caractères.
    - Introduire un seul espace-blanc.
    - Introduire une chaîne complète (nombre maximum autorisé de caractères) d'espaces-blancs.
    - Introduire un seul '-'.

- Test des combinaisons d'erreur : voir comment le logiciel réagit lorsque plusieurs erreurs sont produites simultanément
  Exemple : (Projet AE– gestion de la journée d'entreprises)
  Laisser le champ « nom de l'entreprise » et le champ « adresse » non remplis.

- Test de données erronées dans un input.

- Test des recherches.
  - Critères ne correspondant à aucune donnée.
  - Critères correspondant à un nombre trop élevé de données (ne pas afficher plusieurs milliers de données dans un tableau).

6. Formalisme

Il faut décrire pas à pas toutes les actions que le testeur devra exécuter et pour chacune d'elles, le résultat attendu.

Il est essentiel de décrire le résultat attendu de chaque action de manière exhaustive de sorte de détecter toute anomalie lors des tests (y compris anomalies de régression).

Puisque le testeur devra documenter l'exécution des tests, il serait judicieux de prévoir deux cases « ok » et « ko » ainsi qu'une zone de commentaires.

Voir TESTFONCT_0002-CreationPartenaire.docx

7. Description des données d'output

Il faut enfin décrire le résultat attendu global en fin d'exécution du scénario. Il faut documenter toutes les données présentes dans l'application à la fin du test.

Exécuter les tests

Le document de tests est donc rempli lors de chaque exécution. Il est enregistré afin de pouvoir être consulté ultérieurement.

Les résultats des tests sont analysés et des statistiques sont produites afin de vérifier que le développement du logiciel s'améliore. Si cela est nécessaire, les méthodes de travail de l'équipe seront améliorées pour augmenter la fiabilité du logiciel.