

Vérifier et valider une solution

Business Analyse

Analyse Compétences : 3

Ne pas confondre vérification et validation d'un projet. Le premier s'assure que la conception a été bien conçue et réalisée sans erreur, tout en suivant scrupuleusement le cahier des charges. Le deuxième veille que la solution mise en place réponde aux problèmes de base.

Publié mercredi 21 juillet 2021, 15h02
Modifié lundi 26 août 2024, 10h04



By Olivier Paudex

Introduction

La réalisation d'un projet passe par différentes étapes qui vont de la planification à l'intégration de la solution, en passant par l'analyse des besoins et la gestion des exigences. Mais une fois cette dernière livrée, on pourrait croire que le travail du BA s'arrête là. Il n'en est rien, le BA va participer activement à la vérification et la validation de la solution mise en place avec ce qui a été défini dans les exigences. Voilà un petit tour d'horizon des tâches.

La SE (Solution Evaluation)

La **SE** est la section concernant l'évaluation d'une solution, l'analyse de sa performance, et les recommandations pouvant améliorer sa plus-value comme vue par le **BABOK**.

Cette publication va traiter des tests effectués tout au long d'un projet.

Un **PDF** joint à cette publication, sous forme de flux de données, résume toutes les tâches, leurs descriptions, les outils, les parties prenantes impliquées, ainsi que des techniques mnémoniques. Il a été réalisé dans un but pédagogique.

Savoir différencier les deux tâches

La vérification, c'est d'assurer que la conception est bien conçue et sans erreur. La validation veille, quant à elle, que la solution mise en place réponde au besoin ou au problème de base.

Les deux tâches vont de concert. Le cycle démarre souvent quand le client demande de résoudre un problème. L'équipe de projet va alors définir et vérifier que les exigences découvertes résolvent le problème. Les questions à se poser pourrait-être :

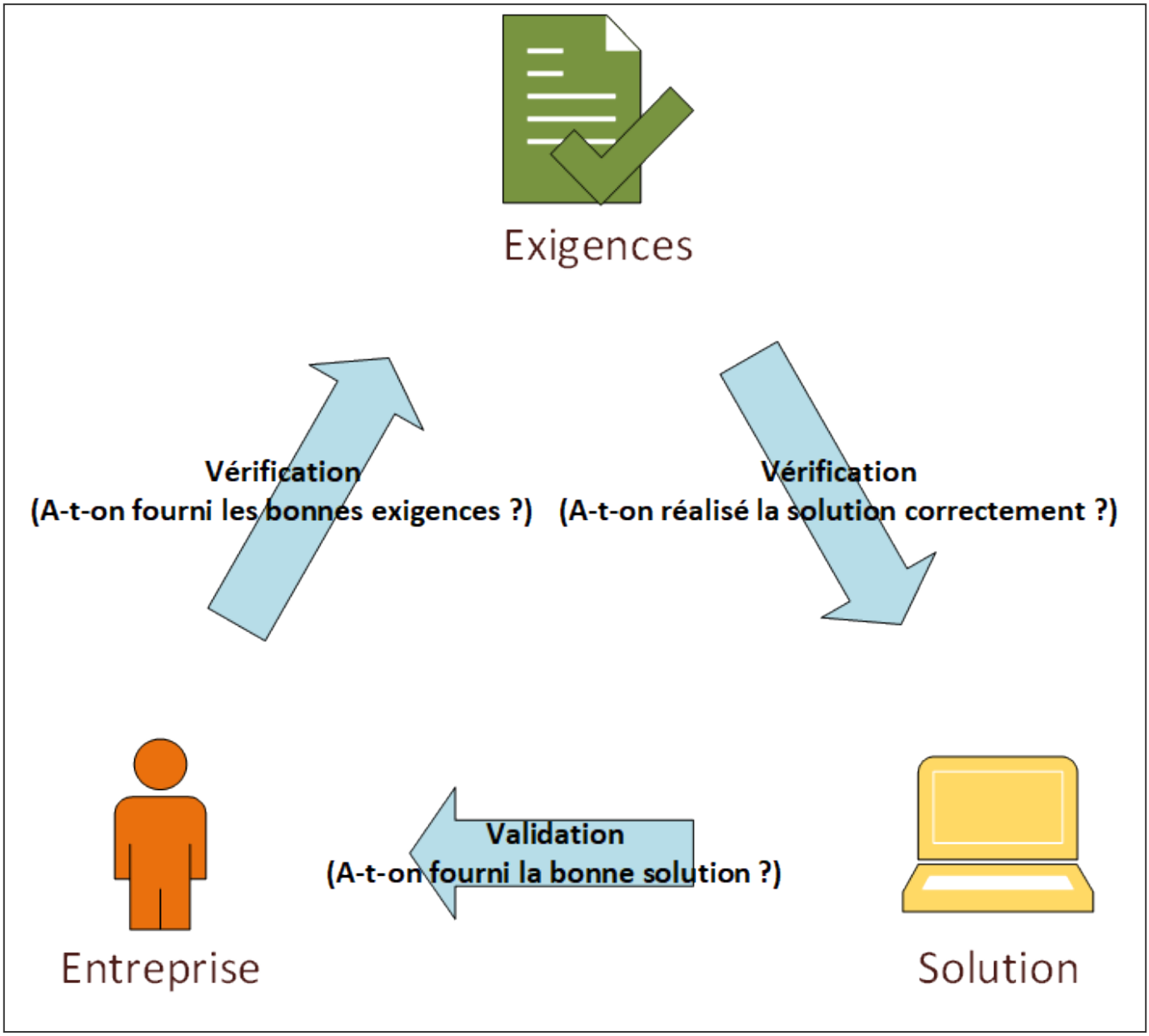
- **A-t-on fourni les bonnes exigences ?**
- Les livrables sont-ils appropriés ?

On passe alors au développement d'une solution et lors de la vérification de la solution, la question à se poser pourrait-être :

- **A-t-on réalisé la solution correctement ?**
- A-t-on résolu le problème principale (root cause) ?

Vient ensuite alors toute une phase de tests, de correction, jusqu'au moment où le client**valide** la solution. La question serait alors :

- **A-t-on fourni la bonne solution ?**
- Est-ce que la solution fournie fonctionne ?



Le cycle de vérification et de validation

Quand débute la phase des tests

Les tests se font dès la phase de planification en ajoutant une estimation du temps nécessaire à effectuer ceux-ci.

Puis dans la phase de recherche, pendant l'élicitation des parties prenantes et l'écriture des exigences dans le **backlog**, une équipe assurant la qualité va les relire et valider si ces dernières sont testables. Le BA est

responsable de décrire tout un arsenal d'occurrences de tests à effectuer. Les tests se terminent par l'acceptation finale de ceux-ci par le client. C'est ce qu'on appelle, en anglais, **un UAT (User acceptance test)**.

Les quatre phases de tests

Le test de fumée (smoke test)

Il permet de vérifier que le solution ne présente pas de gros défaut apparent. Il révèle les simples erreurs qui empêcherait l'exécution des trois autres phases de tests.

Cette définition vient du domaine de l'électronique. Quand une nouvelle machine était mise sous tension pour la première fois, les concepteurs vérifiaient que celle-ci ne **"fumait pas"**, dans le sens propre du terme.

Le test unitaire

C'est la deuxième phase de test, après le test relativement simple de la fumée. C'est un test effectué sur un module ou une petite partie de la solution.

Par exemple, un logiciel est découpé en fonctions qui sont testés les unes après les autres.

L'avantage de soumettre les modules au test unitaire sont les suivants :

- Tant qu'un module n'a pas passé le test unitaire, il ne pourra pas être soumis dans le test d'intégration globale. Par exemple, on va tester le moteur d'une voiture à vide, avant de le monter sur le châssis et de vérifier que celle-ci peut rouler.
- Il est tout à fait possible de hiérarchiser les phases de tests. Pour reprendre l'exemple de la voiture, il est tout à fait possible de vérifier que certains éléments du moteur fonctionne entre eux avant de construire le moteur dans son entier et de l'intégrer au châssis.
- Quand un module est actionné par une interface, il est recommandé de le faire vérifier par plusieurs personnes. Chaque être humain réagit et pense différemment. Faire ainsi permet d'éviter les erreurs que les concepteurs n'auraient pas vues uniquement parce que rien ne suggérait que quelqu'un actionnerait l'interface d'une manière différente.

Le test d'intégration globale

La troisième phase de test consiste à rassembler tous les modules contrôlés dans la deuxième phase et d'éprouver leurs fonctionnements en commun dans un environnement dédié. On appelle ceci, en anglais, un **"Sandbox"**.

Le système de test

C'est la quatrième et dernière phase. C'est aussi le test de la dernière chance laissé aux concepteurs avant que la main passe aux utilisateurs pour la phase finale d'acceptation (**UAT**).

Ce test va confirmer que la solution couvre le problème de départ. Il est catégorisé en plusieurs sous-catégories dont :

La validation des exigences

Ce test vérifie la logique de la solution en s'assurant que les exigences analysées tout au long du projet soient respectées.

Le test de régression

Ici, il s'agit de pouvoir re-tester tous les changements effectués à partir d'un certain point. Ce contrôle permet d'être certain qu'une modification de dernière minute n'interfère pas avec ce qui fonctionnait précédemment.

Le test dynamique

Ce test se déroule la solution doit fonctionner sous différentes circonstances. On peut citer :

- **Le test de performance**. Ceci démontre à quelle vitesse la solution doit réagir. Il doit correspondre au temps de réponse défini dans les exigences non-fonctionnelles.
- **Le test de stress**. Ceci démontre les limites de la solution. Un moteur ne va pas réagir de la même manière sur une autoroute que sur une route de montagne.
- **Le test de volume**. Ceci démontre le bon fonctionnement du système si celui-ci doit traiter une énorme quantité de données. Il est réalisé en prévision du futur.

Le test de sécurité

Ici, il ne s'agit ni plus ni moins de gérer les droits d'accès dans un logiciel ou des fonctions d'une solution qui aurait plusieurs niveaux d'utilisation.

Le test d'installation et de configuration

Ce test assure que le logiciel va s'installer correctement sur différentes plateformes (version, mémoire minimum, etc…), ou fonctionner sur différents navigateurs.

Le test d'utilisation

Ce test demande le retour des utilisateurs quant à l'utilisation de l'application. On cherche à connaître l'avis des utilisateurs sur la facilité d'utilisation de l'interface.

Voici quelques conseils lors de la création d'interfaces :

- **Toujours afficher le statut du système**: Intégrer une barre de progression dans l'interface.
- **Écrivez dans un vocabulaire compréhensible par l'utilisateur**: Faîtes en sorte que le vocabulaire est à la portée de tout le monde.
- **Ajoutez des options d'annulation et de répétition de la dernière tâche effectuée**: Faîtes en sorte que l'utilisateur ait toujours une possibilité d'annuler ou de répéter la dernière tâche effectuée.
- **Soyez consistant** : Utilisez toujours les mêmes mots, les mêmes couleurs, les mêmes symboles.
- **Proposez des listes déroulantes** : Ajoutez des listes à choix, plutôt que de laisser l'utilisateur saisir une information. Cela évite des fautes de frappe.

- **Contrôlez la saisie immédiatement** : Contrôlez par exemple qu'un numéro de téléphone soit bien composé uniquement de chiffres, mais pas de lettres.
- **Offrez des raccourcis pour gagner du temps**: Pensez aux experts ou aux utilisateurs qui vont devoir saisir un grand nombre de fois la même information.
- **Créez une interface intuitive et simple**: N'ajoutez pas d'informations inutiles.
- **Créez des bulles d'informations pour guider l'utilisateur dans la démarche**: Pensez à ajouter de l'aide pour aiguiller l'utilisateur dans son travail.

Définir des occurrences de test

Les occurrences de test sont des instructions pas-à-pas. Pour créer un bon exemple de simulation, il faut entre autres :

- Décrire précisément ce que doit réaliser le testeur.
- Définir ce que le testeur doit entrer comme données et ce qu'il devrait avoir comme résultat.
- Dans quel environnement le test doit s'effectuer (OS, Version, etc...)
- Décrire toutes procédures spéciales ou demandant des manipulations adéquates.
- Documenter la priorité des tests. Est-ce qu'un test doit être effectuer avant un autre.
- Demander au testeur d'approuver le résultat.

Définir un plan de test

Le plan va décrire toutes les étapes de vérification et de validation. Il va décrire comment une solution doit être contrôlée. Voici quelques conseils :

- Expliquez l'objectif du projet et ce à quoi la solution va être soumise.
- Définissez l'environnement dans lequel les tests doivent être effectués.
- Définissez les occurrences qui vont être testés.
- Définissez les données nécessaire pour le test.

Occurrence à tester	Résultat attendu
Acheter un produit taxé	La taxe doit s'ajouter au montant du produit lors de l'achat
Acheter un produit non taxé	Le montant du produit est calculée sans ajout d'une taxe
Acheter plusieurs produits, certains taxés, d'autres non taxés	La taxe doit s'ajouter uniquement au montant du produit taxé

Description des fonctionnalités à tester

- Définissez le cycle de test (temps entre chaque test) et les outils à utiliser.
- Définissez les critères de réussite et d'échec.
- Définissez ce qu'il faut faire en cas d'échec d'un test.
- Déterminez un tableau des testeurs et des responsabilités.

Conclusion

La phase de test est d'une importance capitale. Elle garantit que la solution mise en œuvre fonctionne correctement selon les exigences fournies dans le backlog. La tâche du BA sera de définir un plan de test, dans lequel toutes les occurrences à tester seront décrites. Il est recommandé de découper le plan de test en plusieurs parties et d'effectuer des contrôles unitaires avant de tester la solution dans son ensemble. Il est également recommandé de créer une plateforme de test (**staging**) ou un environnement sans risque (**sandbox**). Une équipe d'assurance qualité va alors exécuter les occurrences de test. Pour terminer, les utilisateurs finaux devront valider le plan en le signant.