Chapitre 3
Exécution des tests

1. Objectif

L'objectif de l'exécution des tests est de dérouler les cas de tests préparés pendant la phase en amont de conception et de préparation. Il s'agit de vérifier que le résultat obtenu dans le logiciel ou l'application est conforme au résultat attendu défini dans le cas de test

Lorsque le résultat obtenu est différent du résultat attendu, les écarts sont analysés et, si l'anomalie est avérée, un ticket anomalie doit être créé pour tracer et suivre le traitement de l'anomalie.

La phase d'exécution est une étape clé, attendue par les équipes projet, car c'est à ce moment que l'application est confrontée à la réalité. Les résultats des tests permettent de valider son bon fonctionnement ou de révéler des anomalies. Ces dernières sont ensuite suivies et traitées à l'aide d'outils tels que HP ALM, Jira ou Squash.

2. Campagne de tests

Préparation

Une campagne de tests est un ensemble de tests, sélectionnés et regroupés pour être exécutés sur une période donnée pour une version de l'application.

Chaque campagne de tests a des exigences en termes d'objectifs et en termes de délai ou de durée. L'objectif d'une campagne de tests peut être :

– Tester les nouvelles fonctionnalités du logiciel.
– Tester la non-régression suite à une nouvelle mise à jour ou évolution du logiciel.
– Tester le bon fonctionnement du logiciel suite à une migration de serveur de données.

En fonction de l'objectif de la campagne de tests, le testeur va venir « piocher » dans le référentiel ou le patrimoine de tests préparés. Les tests à intégrer correspondent à la préparation de la campagne de tests.

Avant de démarrer l'exécution des tests, il faut préparer une campagne de tests.

Des outils de gestion de tests comme HP ALM et Jira Xray permettent de créer et gérer les campagnes de tests et de suivre les résultats des tests.

Le testeur est responsable de la création et de l'organisation de sa campagne de tests, le chef de projet test peut approuver si le périmètre de test est complet et si les priorités respectées.

La préparation de la campagne de tests est une activité organisationnelle essentielle. Elle permet de suivre les résultats des tests pour une version donnée de l'application et un objectif précis, tout en assurant le suivi de l'évolution de la qualité du logiciel d'une version à l'autre, grâce à l'historisation des campagnes et de leurs résultats.

L'exécution de tests se fait sur l'environnement de test dédié. S'assurer de la disponibilité et de la bonne configuration de l'environnement de test fait partie de l'étape préparatoire de l'exécution des tests.

Exécution

L'exécution d'une campagne de tests consiste à dérouler un à un les cas de test prévus, en enregistrant progressivement les résultats d'exécution de chaque test, voire de chaque étape lorsqu'ils sont détaillés.

Voici le résultat que peut avoir un test pendant à son exécution :

- Statut « Ok » ou « Passant » selon l'outillage et la nomination choisis : lorsque le résultat obtenu dans l'application est conforme au résultat attendu.
- Statut « KO » ou « En échec » : lorsque le résultat obtenu dans l'application est différent du résultat attendu.
- Statut « Bloqué » : lorsque l'exécution d'un test est empêchée par une anomalie existante, et en raison de dépendances, peut bloquer et affecter d'autres tests.

L'exécution de la campagne de tests comprend également la remontée des écarts identifiés sous forme d'anomalies, le suivi de leur traitement, ainsi que le re-jeu des tests en échec une fois les anomalies corrigées.

Nous détaillerons dans le chapitre suivant la gestion des anomalies qui est une partie intégrante de l'exécution des tests.

3. Gestion des anomalies

3.1 Objectifs

Nous avons vu dans les chapitres précédents qu'une anomalie est un écart entre un résultat attendu et un résultat obtenu détecté sur les environnements de test. Une anomalie est aussi appelée bug ou défaut, on utilise plus souvent le terme anomalie.

Nous appelons les anomalies détectées en production des incidents.

La gestion des anomalies est une étape importante dans le processus de test et de développement. La bonne gestion du portefeuille des anomalies ou ce qu'on appelle aussi backlog des anomalies doit se faire via une méthodologie et un processus bien défini.

La gestion des anomalies commence depuis la détection de l'anomalie jusqu'à son traitement et sa clôture. Une anomalie demeure active tant qu'elle n'est pas corrigée, validée et clôturée.

C'est pour cela qu'une bonne gestion des anomalies tant sur le plan opérationnel que managérial est un facteur clé pour garantir une bonne qualité des livraisons et un pilotage avec des indicateurs fiables et efficaces.

Hormis le respect des critères d'acceptance et la satisfaction des utilisateurs, qui reste l'objectif final et collectif, une bonne gestion des anomalies permet la détection des actions correctives et aussi préventives dans le cadre d'une amélioration continue des livrables, des processus et des bonnes pratiques de tests et de développement afin de capitaliser d'une livraison à une autre.

Une bonne gestion des anomalies facilite également l'organisation et la priorisation des travaux des testeurs et développeurs et facilite surtout la communication et la collaboration entre ces deux équipes.

3.2 Processus de traitement des anomalies

Comme évoqué dans la section précédente, la gestion d'une anomalie commence depuis sa détection jusqu'à sa fermeture. Entre les deux états, l'anomalie passe par différentes étapes : c'est ce que l'on appelle le cycle de vie de l'anomalie, en anglais *workflow* de l'anomalie.

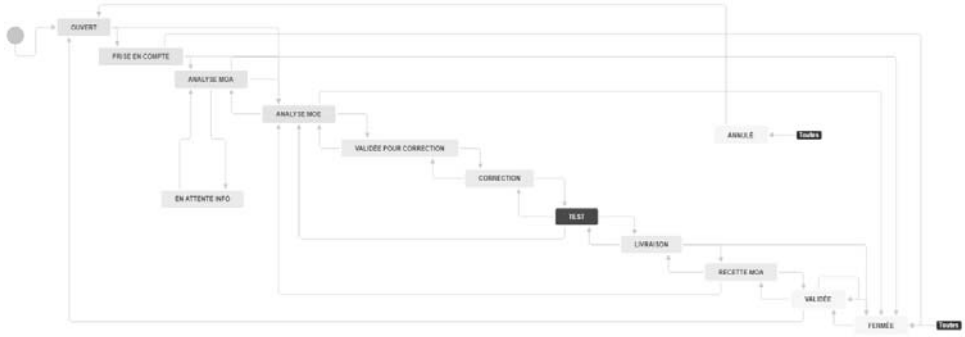
Le cycle de vie d'une anomalie est défini par chaque organisation et formalisé dans la stratégie de tests et parfois plus en amont dans le document PAQ (plan d'assurance qualité).

Le cycle de vie d'une anomalie peut varier en fonction de l'environnement dans lequel elle a été détectée.

Remarque

Une anomalie détectée sur l'environnement de validation aura un cycle de vie avec plus ou moins d'étapes qu'une anomalie détectée sur l'environnement recette utilisateur.

Exemple de workflow



Voici les principales étapes du cycle de vie d'une anomalie.

3.2.1 Détection de l'anomalie

L'anomalie une fois détectée doit être signalée par le testeur au développeur. Une fiche anomalie doit alors être créée en respectant les bonnes pratiques de signalement d'une anomalie. Selon l'outillage utilisé, des champs obligatoires doivent être renseignés pour que la fiche anomalie soit exploitable par l'équipe de développement.

Voici les principales propriétés d'une anomalie :

- **Le titre** : le titre de l'anomalie doit être clair, signifiant et intégrer le nom de la fonctionnalité en échec pour faciliter l'affectation de l'anomalie au bon développeur, c'est-à-dire au développeur ayant travaillé sur l'implémentation de la fonctionnalité concernée ou au développeur maîtrisant le domaine fonctionnel impacté.
- **Les étapes ou le scénario de reproduction** : il s'agit de renseigner toutes les étapes du test ayant permis de détecter l'anomalie. Il faut que les étapes soient complètes et claires en mentionnant, pour chaque étape, les données de tests utilisées ainsi que le résultat attendu et le résultat obtenu. Le développeur doit avoir toutes les informations nécessaires pour qu'il puisse reproduire l'anomalie sur son environnement de développement.

Un manquement dans les étapes de reproduction entraînera un temps supplémentaire d'aller-retour entre le testeur et le développeur ou un rejet de l'anomalie par l'équipe développement. Dans les deux cas, il en résulte un délai supplémentaire évitable pour le traitement de l'anomalie ainsi qu'un temps consommé par le testeur au détriment d'autres tâches.

– **La sévérité** : la sévérité d'une anomalie est une mesure de l'impact qu'une anomalie a sur le fonctionnement d'un système ou d'un processus. Elle aide à déterminer la gravité et l'importance de sa résolution.

Les critères communs pour évaluer la sévérité d'une anomalie sont :

– **Impact sur le fonctionnement** : est-ce que l'anomalie empêche le système de fonctionner correctement ou le rend inexploitable ?
– **Pertes financières** : quel est le coût associé à l'anomalie, en matière de pertes financières directes ou indirectes ?
– **Sécurité** : l'anomalie présente-t-elle un risque pour la sécurité des utilisateurs ou du personnel ?
– **Réglementation et conformité** : l'anomalie entraîne-t-elle des violations de normes ou de régulations qui pourraient entraîner des sanctions ?
– **Effet sur les utilisateurs** : quel est l'impact sur l'expérience utilisateur ? L'anomalie cause-t-elle des désagréments mineurs ou des interruptions majeures ?

Voici des exemples de catégories de sévérité d'une anomalie souvent utilisées :

– **Bloquante** : une anomalie est dite bloquante si elle est liée à un processus métier critique, c'est-à-dire que, si elle survient en production, l'utilisateur sera bloqué dans ses fonctions, ou que l'utilisation du système sera bloquée. Une anomalie peut aussi être taguée comme bloquante si elle bloque l'avancement des tests avec les testeurs bloqués dans leurs travaux.
– **Majeure** : une anomalie est dite majeure si elle est liée à une fonctionnalité majeure et que le système peut continuer à fonctionner.
– **Mineure** : une anomalie est dite mineure lorsqu'elle cause des désagréments mineurs sans impacter ou bloquer le fonctionnement de la fonctionnalité.

Remarque

La définition de chaque sévérité est propre à chaque organisation. Cependant, il est important que la définition soit formalisée et partagée avec l'ensemble des parties prenantes.

– **Priorité** : nous avons vu précédemment la notion de sévérité, qui est l'évaluation de la gravité de l'anomalie sur le système et les utilisateurs. La priorité va venir déterminer l'urgence avec laquelle l'anomalie doit être traitée. Il arrive qu'on ait des anomalies avec la même sévérité, mais de priorité différente. La combinaison des deux va permettre de mieux prioriser le traitement des anomalies.

Prenons un exemple d'application pour mieux comprendre :

- Anomalie 1 : une faille de sécurité critique dans un logiciel bancaire.

Sévérité : bloquante (risque de pertes financières massives et impact sur la sécurité).

Priorité : maximale.

- Anomalie 2 : un bug graphique mineur dans une application de gestion de projet.

Sévérité : mineure (impact visuel mineur, aucune perte fonctionnelle).

Priorité : basse

En évaluant le couple sévérité/priorité, les équipes de développement et de maintenance peuvent prioriser efficacement les anomalies à corriger, en tenant compte de leur gravité et de leur urgence.

La priorité d'une anomalie peut-être évaluée en fonction des éléments suivants :

– **Version** : il s'agit de la version applicative du logiciel.

  Chaque nouvelle livraison équivaut à une nouvelle version. Il est important de maintenir la bonne pratique en développement d'incrémenter les versions applicatives à chaque nouvelle livraison. Dans la fiche anomalie, on utilise deux types de versions. La version dans laquelle l'anomalie a été détectée et la version dans laquelle l'anomalie a été corrigée et validée.

  Cette information permet de mieux maîtriser le contenu de chaque livraison. Dans le cas d'un blocage, il sera plus simple d'identifier la livraison ayant généré le blocage et de pouvoir effectuer un retour arrière, restaurer l'application dans la version antérieure et assurer la continuité de service. Les corrections pourront se poursuivre alors dans un autre environnement, notamment celui du développement sans impact.

Prenons l'exemple d'une anomalie détectée dans une version V1, corrigée dans une version V2 et réapparue dans une version V3. Si le versioning est bien configuré et son renseignement respecté dans la fiche anomalie, on peut facilement constater qu'il s'agit d'une régression et évaluer ensuite la cause racine pour mieux prévenir et anticiper les régressions dans les versions futures.

– **Environnement** : il s'agit de l'environnement dans lequel l'anomalie a été détectée. Cette information est importante pour que le développeur puisse reproduire l'anomalie dans les mêmes conditions que celles du testeur. Le correctif doit être livré sur l'environnement de détection de l'anomalie et, si validé, il doit être propagé par le développeur sur les autres environnements.

– **Rapporteur** : il s'agit de renseigner le nom du testeur qui a détecté l'anomalie. Cela permettra au développeur de prendre contact directement avec le testeur en cas de question ainsi que de le notifier une fois l'anomalie corrigée.

– **Responsable** : il s'agit de renseigner le nom du développeur ou du point d'entrée de l'équipe développement en fonction du processus défini.

– **Commentaire** : dans la majorité des outils de gestion des anomalies disponibles sur le marché, il existe un champ commentaire dans la fiche anomalie.

  Il est recommandé d'utiliser ce champ pour tracer les échanges avec les développeurs ou les autres intervenants dans le traitement de l'anomalie.

Il est aussi recommandé de demander au développeur, pour chaque correction, de renseigner dans ce champ la cause identifiée et le résultat de son analyse pour que cela serve de base pour les prochaines corrections.

3.2.2 Assignation de l'anomalie

L'assignation de l'anomalie c'est son affectation à la personne ou à l'équipe en charge de son traitement. C'est le processus de traitement des anomalies qui définit à qui affecter l'anomalie, et dans quel cas, ainsi que l'état ou le statut de l'anomalie associé.

Une anomalie liée au code peut être assignée au développeur directement, si le testeur connaît le nom du développeur ayant travaillé sur l'implémentation de la fonctionnalité impactée. Sinon, dans d'autres cas, l'anomalie est assignée à un point d'entrée identifié qui peut être aussi le responsable de l'équipe développement. C'est lui qui se charge de répartir les anomalies en fonction des compétences et de la disponibilité de chaque développeur.

Si l'anomalie est de type documentaire par exemple, elle est assignée à l'équipe des *business analysts*.

Remarque

L'affectation d'une anomalie se fait en fonction de sa typologie et du processus de traitement des anomalies définies.

3.2.3 Traitement de l'anomalie

Le traitement de l'anomalie consiste à la prise de connaissance de son contenu par l'équipe concernée.

Si c'est le développeur qui prend connaissance de l'anomalie, il va tout d'abord essayer de la reproduire sur son environnement de développement en local. D'où l'importance, comme soulevé dans la section précédente, de bien décrire les étapes du scénario ayant permis de détecter l'anomalie. Si l'anomalie est avérée, il procédera à sa correction. Si l'anomalie n'est pas reproductible en local, le développeur fera une analyse plus poussée ou retournera l'anomalie au testeur.