Polarion ALM : un chef d'orchestre pour gérer les exigences et les tests logiciels

La prise de conscience de l'importance des tests logiciels a émergé vers la fin des années 1960 suite à ce qui a été nommé « la crise du logiciel », qui a mis en évidence les difficultés rencontrées pour satisfaire au fameux triptyque « coût, délai, qualité » des projets. (Notons que derrière Qualité se cachait aussi la Conformité, à savoir l'adéquation aux exigences exprimées ou réglementaires). Près de 50 ans après, même si la typologie de tests n'a pas profondément changée (unitaire, intégration, système, acceptation, etc.), l'avènement des Frameworks de tests unitaires, de l'intégration continue, de nouvelles approches (BDD Behavior Driven Development), les contraintes de l'automatisation des tests au niveau des IHMs, et enfin des pratiques Agile ont permis d'inverser « la pyramide des tests », son socle n'étant plus aujourd'hui les tests dits fonctionnels, mais les tests unitaires, avec une automatisation poussée des couches basses.

Avant de rentrer dans le vif du sujet des « tests logiciels », nous définissons et exécutons des tests pour délivrer des solutions de qualité, répondant aux besoins des futurs utilisateurs et clients. Cette recherche de qualité passe en premier lieu par une gestion efficace des exigences, ne dit-on pas que parmi leurs critères d'acceptation, qu'elles doivent être vérifiables, mesurable ou encore traçable ?

Une couverture globale du cycle de conception du logiciel

Ce caractère indissociable entre les artefacts exigences et tests rend nécessaire de s'appuyer sur un chef d'orchestre et c'est exactement le rôle que joue **Polarion ALM** dans une chaîne de développement logiciel.

Polarion ALM est une solution unifiée de gestion du cycle de vie des applications et systèmes, s'appuyant sur un référentiel centralisé, prônant la collaboration entre toutes les parties prenantes (comment, un testeur peut échanger avec un développeur ou un analyste métier ?), le pilotage des processus (ou comment guider un acteur dans son workflow), avec cette flexibilité permettant de gérer de manière efficiente et cohérente tous types d'artefacts, nommés **Work Items** : Que ce soit des Work Items de type exigence, user story, cas de test, plans de test, ou défaut, les mêmes règles et principes s'appliquent pour notamment :

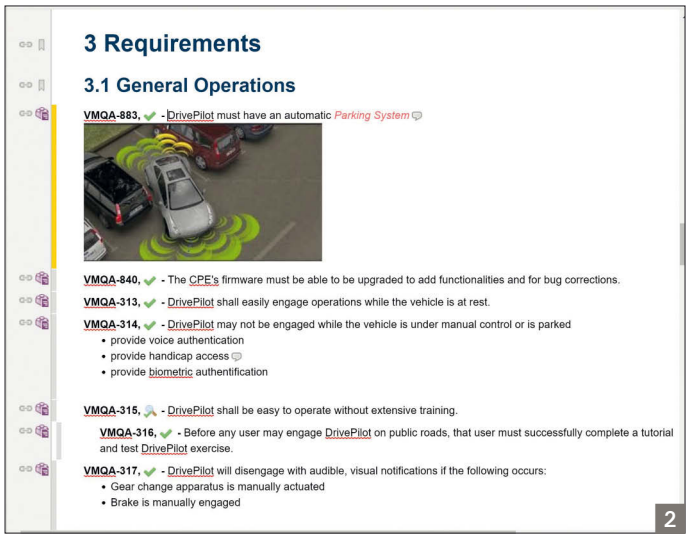
- Spécifier les exigences et artefacts de tests dans des documents LiveDoc collaboratifs, versionnés, historisés, facilitant les revues et pouvant être signés électroniquement,
- Analyser l'impact des changements et le taux de couverture par la traçabilité entre les exigences et les tests,
- Délivrer des métriques sur l'ensemble des artefacts du projet.

Polarion ALM est une solution reconnue et largement déployée pour spécifier, documenter et approuver les exigences dans les secteurs du développement logiciel, des systèmes embarqués, ou dans des domaines fortement réglementés comme les dispositifs médicaux ou l'avionique. 1 2

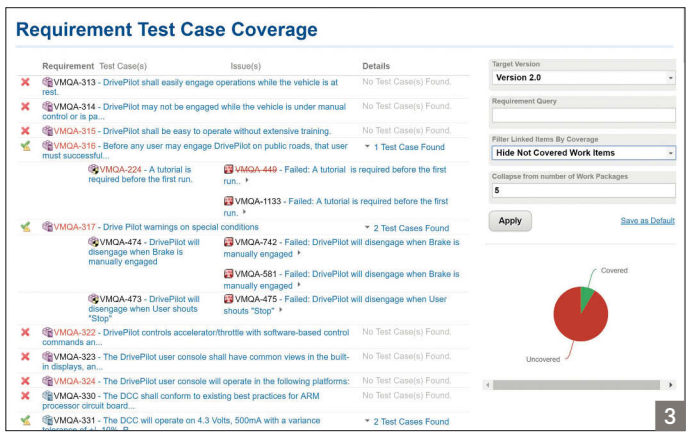
L'utilisation de Polarion ALM pour aligner les tests et les mesurer en regard des exigences, couvrir les activités propres aux tests logiciels comme définir la stratégie de test, organiser et définir les artefacts de tests, piloter leur exécution, consolider les informations et apporter de la visibilité sur l'ensemble – avancement, résultats, conformité –, permet de faciliter la continuité entre les différentes phases des projets : Il n'y a pas de rupture, les utilisateurs communiquent autour d'un langage commun, une même interface utilisateur, avec une navigation aisée entre les différents Work Items par la traçabilité, une visibilité « sur mesure » en fonction des profils (ex : un analyste métier pourrait lire les plans de tests, sans pouvoir les modifier, et inversement, un Testeur peut consulter voire commenter les documents de spécification, ce qui est fort utile pour son métier, mais sans pouvoir les mettre à jour). 3

Comment les choses se déroulent-

1



2



3

elles quand les tests sont spécifiés dans un environnement différent que celui des exigences ? Nous voyons alors les exigences être copiées dans l'environnement de test. Ces exigences dupliquées sont alors associées aux cas de tests qui seront alors exécutés avec des résultats stockés dans l'environnement de test. Alors certes des intégrations sont possibles et existent entre Polarion ALM et d'autres gestionnaires de tests, mais par exemple, comment et où se mesure la couverture des exigences initiales par les tests, que ce soit en termes de définition ou d'exécution, dans l'outil de gestion des exigences ou l'outil de tests ? Comment communiquer et gérer aisément le workflow d'une user story qui passe en état final « *verified done* » une fois les tests validés par les acteurs QA ? Des difficultés parmi d'autres qui sont facilement levées quand les barrières entre activités du génie logiciel sont ouvertes.

Dans la pratique, le testeur va pouvoir rédiger sa spécification de tests dans un document LiveDoc, valoriser les propriétés du test (sévérité, type de test, etc.), relier ces tests aux autres Work Items (exigences, tâches, fonctions, etc.) par lien de traçabilité adéquat. 4

Cette approche pour rédiger les plans et cas de tests est novatrice en permettant donc de s'appuyer sur ce format documentaire LiveDoc, alors que généralement, les tests sont décrits uniquement dans une structure tabulaire hiérarchisée. Avec Polarion ALM, les utilisateurs bénéficient des deux vues systématiquement, il n'y a pas de double-saisie dans une représentation document, puis dans une représentation table. Un utilisateur peut basculer en un clic d'une vue à l'autre. Procédant ainsi, il est possible d'appliquer aux spécifications des tests les mêmes bonnes pratiques que pour les spécifications des exigences, à savoir élucidation, documentation, revue par les pairs, approbation ou refus de la spécification, et signature électronique si requise par la réglementation. 5

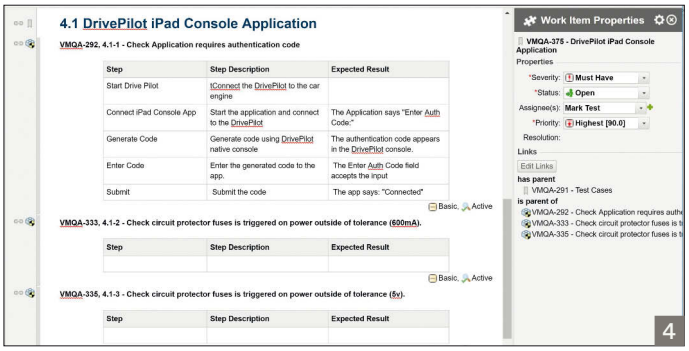
Donner de l'importance au processus de test

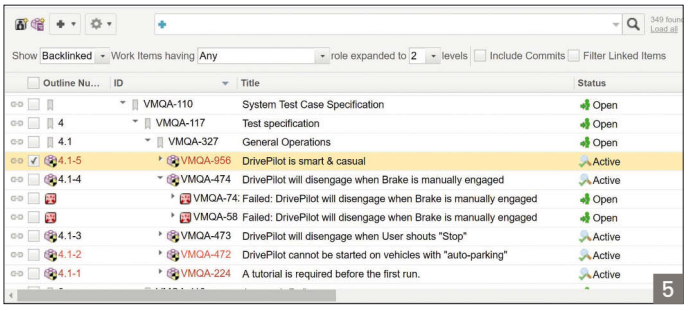
Pour les utilisateurs habitués aux gestionnaires de test « traditionnels », qui peuvent être dirigistes dans la manière de décrire les tests, d'exécuter les campagnes, et mesurer l'avancement ou la couverture, Polarion ALM peut dérouter au premier abord car il donne de la latitude aux administrateurs de la plate-forme pour mettre en place un processus de test ad-hoc, sur mesure : Il n'y a pas qu'un seul type de cas de test, pas un seul workflow, ni une seule approche pour analyser les résultats des campagnes. Polarion ALM offre la souplesse pour concevoir le modèle de données et le processus adapté à l'organisation et au cycle de vie des projets. Ce workflow sur mesure peut être couplé à la signature électronique des résultats de tests pour répondre notamment à la norme médicale 21 CFR Part 11. L'*Audit Trail* sous Polarion permettra de retracer et consulter l'ensemble de l'historique propre aux tests, aux exigences et user stories vérifiées, aux anomalies générées ou même au code source impacté. 6

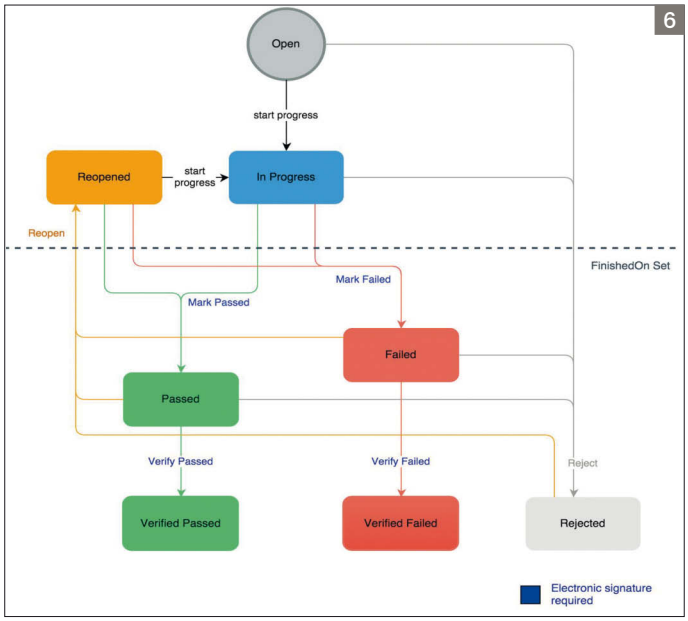
Piloter les campagnes de tests manuels

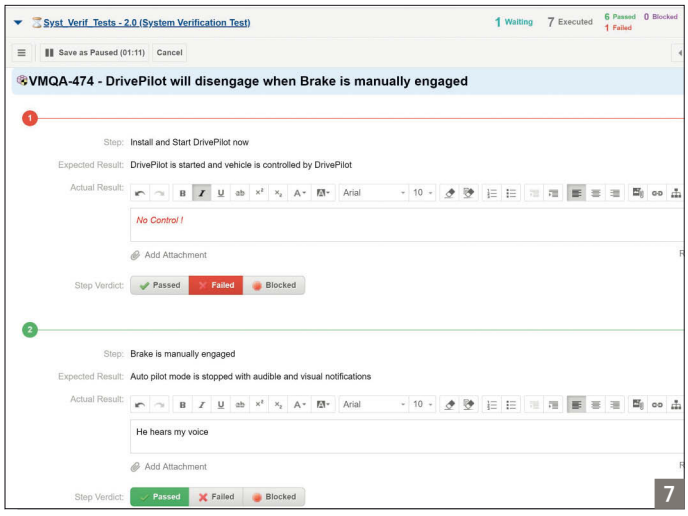
Même si une automatisation plus poussée des tests est visée pour les tests unitaires, les tests d'acceptation ou la couverture de l'*Happy Path*, l'exécution des tests de recette est encore souvent manuelle. Ici encore le focus est mis sur l'adaptabilité. Pour sélectionner les cas de tests à exécuter dans un *Test Run* pour un nouveau *build* du logiciel, l'approche la plus intéressante est de définir un modèle qui va permettre d'alimenter automatiquement ce *Test Run* à partir du contenu d'un document de spécification de tests, ou à partir d'une requête. Cette sélection automatisée peut se faire au fil de l'eau : un nouveau test est créé dans le document pendant que les campagnes sont en cours, le ou les *Test Runs* liés à ce document sont dynamiquement mis à jour. Les étapes de tests peuvent également être variabilisés afin de pouvoir couvrir les mêmes cas de tests tout en exploitant des jeux de données différents. 7

La page rapportant de la progression et des résultats des tests donne une vision d'ensemble de l'état de la campagne et est aisément personna-









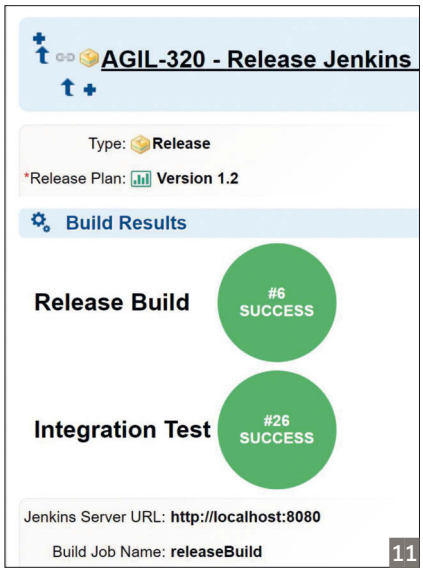
lisable. Dans notre exemple, une fiche d'anomalie a été produite automatiquement et liée au test (dans sa révision d'exécution) et au *Test Run* en échec. Le cycle d'analyse et de résolution de ce défaut constaté sera suivi au niveau de ce Work Item *Defect*, avant de pouvoir reprendre l'exécution. Pour cette gestion des anomalies, vous avez le choix de les gérer sous Polarion ALM, ou dans un autre environnement dédié (notamment Jira pour lequel un connecteur de synchronisation existe). 8

Automatiser les tests dans un pipeline de livraison continue

Quand on automatise les tests, en particulier ceux des niveaux inférieurs de la pyramide (tests unitaires et tests d'intégration), on emploie au-delà des automates eux-mêmes, différentes solutions qui vont nous permettre de mettre en place une chaîne d'intégration continue (CI) et de déploiement continu (CD), par exemple avec des outils comme Jenkins ou GitLab. Polarion ALM se veut par ailleurs agnostique dans le choix des automates retenus. En effet, du fait de la diversité et portée des tests, les entreprises utilisent en moyenne plus de trois outils d'automatisation et d'exécution de ces tests. Les intégrations natives de Polarion et son ouverture via ses Java APIs/Web Services publiés dans le *SDK (Software Development Kit)* permettent de mettre l'outil ALM au cœur de ce framework complet de *Build*, *Test*, *Deploy*, tout en gardant le lien fort avec les exigences ou user stories initiales. 9

Prenons par exemple l'utilisation du serveur d'intégration continue Jenkins : Les résultats des tests automatisés seront remontés sous Polarion sous la forme d'un Test Run, avec une présentation et une consolidation identique comme pour les tests manuels, et ainsi bénéficier d'un environnement unifié pour le suivi des tests. Ce transfert des données vers le gestionnaire de test s'effectue via un fichier résultat produit par l'automate au format xml (format xUnit), ou par une intégration sur mesure via les Web Services. 10

Jenkins est un chef d'orchestre, il peut se suffire à lui-même pour soumettre les jobs. Cependant pour s'affranchir de devoir basculer d'un environnement à l'autre, Polarion ALM montre à nouveau sa flexibilité et son ouverture en permettant de déclencher toute une chaîne de traitement, qui va successivement soumettre les jobs de Builds sous Jenkins, remonter leur statut et générer les Test Runs avec les résultats de tests sous Polarion. 11



Un référentiel centralisé mais ouvert pour accroitre la qualité des logiciels

Tout comme les pratiques Agile qui tendent à rapprocher les différents acteurs impliqués dans la conception logicielle, qu'ils soient Product Owner, développeurs ou testeurs, Polarion ALM se veut être l'environnement central qui va permettre d'unifier et rendre aisément disponibles l'ensemble des données relatives aux activités de tests, et au-delà aux activités de spécifications, de tâches de développement, de gestion des anomalies, etc. Ce lien étroit entre toutes ces données va permettre de faciliter la communication et la collaboration entre les acteurs, de donner plus de visibilité pour une meilleure prédictibilité des releases, de s'affranchir des duplications d'artefacts de tests entre environnements disparates, de disposer des éléments de traçabilité permettant une analyse d'impact plus rapide et plus simple. Dans des frameworks de tests modernes, Polarion ALM ne couvre pas tout cela seul, le logiciel s'ouvre vers l'existant, s'intègre avec les logiciels tiers des pipelines CI/CD et s'adapte aux besoins et visions des architectes logiciels.

•