Stratégie de tests

Table des matières

Section 1 Sommaire de cette formation

Définitions et typologie des tests

Introduction à la qualité Logiciels / coût de la qualité

Types de tests

Méthodologie de tests

Cibles du test : les composants logiciels (fiabilité, performances, utilisabilité)

Étapes du processus de test

Construction de scénarios de test : mise en pratique

Mise en œuvre des tests

Acteurs et leur rôle (informatique, client, sous-traitant, …)

Coordination des tests

Automatisation des tests

Documents de recette

Rédaction d'un cahier de tests

Section 2 La Qualité logicielle – les enjeux

2.1 Définir la Qualité logicielle

https://fr.wikipedia.org/wiki/Qualit%C3%A9_logicielle

https://www.softfluent.fr/blog/qualite-logicielle/

La qualité d'un **matériel** peut s'établir à partir du choix de la qualité de ses composants, de sa conception et réalisation. En moyenne les vices cachés sont peu nombreux, la durée de vie du matériel peut être établie – taux d'usure, le mtbf (*mean time between failures*) peut être calculé.

La **qualité logicielle** est une appréciation globale d'un logiciel, basée sur de nombreux indicateurs.

La **complétude** des fonctionnalités, la **correction et précision** des résultats, la **fiabilité**, la **tolérance de pannes**, la **facilité et la flexibilité** de son utilisation, la **simplicité**, **l'extensibilité**, la **compatibilité et la portabilité**, la **facilité de correction et de transformation**, la **performance**, la **cohérence et l'intégrité** des informations qu'il contient sont tous des facteurs de qualité.

Un logiciel est un produit qui n'a pas une fiabilité prédictible, de plus il ne s'use pas dans le temps.

Donc une anomalie survient ou ne survient pas dans l'exécution du logiciel, l'anomalie est présente de manière latente et peut ne jamais survenir.

La qualité d'un logiciel dépend entièrement de sa construction et des processus utilisés pour son développement, c'est par conséquent un sujet central en génie logiciel.

Une appréciation globale de la qualité tient autant compte des facteurs *extérieurs*, directement observables par l'utilisateur, que des facteurs *intérieurs*, observables par les ingénieurs lors des revues de code ou des travaux de maintenance.

La norme 25010 définit six groupes d'indicateurs de qualité des logiciels :

- la **capacité fonctionnelle**. C'est-à-dire la capacité qu'ont les fonctionnalités d'un logiciel à **répondre aux exigences** et besoins explicites ou implicites des usagers. En font partie la précision, l'interopérabilité, la conformité aux normes et la sécurité ;
- la **facilité d'utilisation**, qui porte sur l'effort nécessaire pour apprendre à manipuler le logiciel. En font partie la facilité de compréhension, d'apprentissage et d'exploitation et la robustesse - une utilisation incorrecte n'entraîne pas de dysfonctionnement ;
- la **fiabilité**, c'est-à-dire la capacité d'un logiciel de rendre des résultats corrects quelles que soient les conditions d'exploitation. En font partie la tolérance aux pannes - la capacité d'un logiciel de fonctionner même en étant handicapé par la panne d'un composant (logiciel ou matériel) ;

- la **performance**, c'est-à-dire le rapport entre la quantité de ressources utilisées (moyens matériels, temps, personnel), et la quantité de résultats délivrés. En font partie le temps de réponse, le débit et l'extensibilité - capacité à maintenir la performance même en cas d'utilisation intensive ;
- la **maintenabilité**, qui mesure l'effort nécessaire à corriger ou transformer le logiciel. En font partie l'extensibilité, c'est-à-dire le peu d'effort nécessaire pour y ajouter de nouvelles fonctions ;
- la **portabilité**, c'est-à-dire l'aptitude d'un logiciel à fonctionner dans un environnement matériel ou logiciel différent de son environnement initial. En font partie la facilité d'installation et de configuration dans le nouvel environnement.

2.2 Coût de la Qualité

https://pyx4.com/blog/4-types-de-couts-de-non-qualite/

Coûts d'Obtention de la Qualité (COQ) = Coûts de la Qualité (CQ) + Coûts de Non Qualité (CNQ)

2.2.1 Les 2 types de coûts de non qualité (CNQ)

2.2.1.1 Les coûts de non qualité internes

Ces coûts regroupent les frais relatifs à un **dysfonctionnement**, à la non qualité d'un **produit ou d'un service qui ne serait pas conforme au standard** ou aux exigences de l'organisation, avant que celui-ci ne quitte l'organisation, c'est-à-dire **avant sa livraison au client**. Entrent dans cette catégorie les rebuts, les coûts de réparation ou de remplacement de machine défectueuse, etc.

2.2.1.2 Les coûts de non qualité externes

Ces coûts regroupent aussi les frais relatifs à un produit ou service non conforme au standard, aux exigences qualité mais **après qu'il a quitté l'entreprise,** c'est-à-dire **après sa livraison au client**. Entrent dans cette catégorie tous les frais qui auraient pu être évités si le livrable avait satisfait le client. Par exemple le **traitement des réclamations client**, le remplacement des produits défectueux ainsi que leur réacheminement, les éventuels pénalités et/ou dommages et intérêts, etc.

2.2.2 Les 2 types de coûts de la qualité (CQ)

Les deux types de coûts de la qualité sont les coûts de prévention et les coûts de détection. La somme des coûts de ces deux catégories représente ces fameux coûts de non qualité (CNQ) que l'on considère à juste titre comme la marge supplémentaire qu'il serait possible de dégager si l'organisation réalisait **une exécution parfaite**, sans aucun dysfonctionnement. Les organisations cherchent tout naturellement à mettre en place un certain nombre de **mécanismes visant à réduire ces coûts de non qualité ou CNQ**.

Ces mécanismes sont considérés comme les **Coûts de la Qualité (CQ)** et sont représentés de la façon suivante :

2.2.2.1 Les coûts de prévention

Ces coûts sont associés aux **investissements de tout type** (humains comme matériels) engagés pour **prévenir et réduire au minimum les dysfonctionnements de l'organisation**. Ils incluent notamment les frais de maintien du système qualité, les activités d'assurance qualité mais aussi toutes les formations ayant pour but de limiter le nombre d'anomalies.

2.2.2.2 Les coûts de détection

Ces coûts sont associés aux dépenses engagées pour **vérifier la conformité du produit aux exigences de qualité**. Il s'agit en d'autre termes des frais générés par la recherche de la non qualité quelle qu'elle soit. D**'identifier les dysfonctionnements internes comme externes.** On retrouve dans cette catégorie les coûts des machines de contrôle ou les charges du service Qualité affectées à cette activité de détection, de contrôle.

Section 3 La documentation projet

3.1 Enjeux

La qualité de la documentation Projet est un des facteurs de réussite du projet.

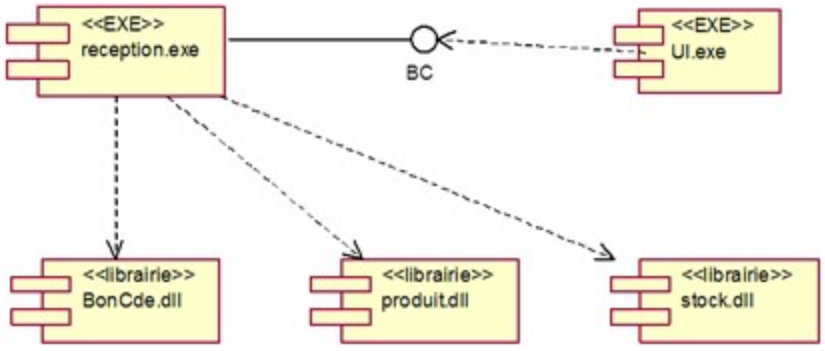
- Un cahier des charges précis et un interlocuteur fiable côté client permettent d'élaborer une spécification système et fonctionnelle de qualité.
- La spécification système de qualité permet d'établir un plan de validation système précis,
- La spécification fonctionnelle permet d'établir un plan de validation fonctionnelle précis, et est un entrant de la conception
- La conception de qualité fournit un support fiable et une autonomie à l'équipe de développement et à l'équipe d'intégration.

Ces documents utilisent des méthodologies connues (UML, description fonctionnelle SADT …) plutôt que « maison ».

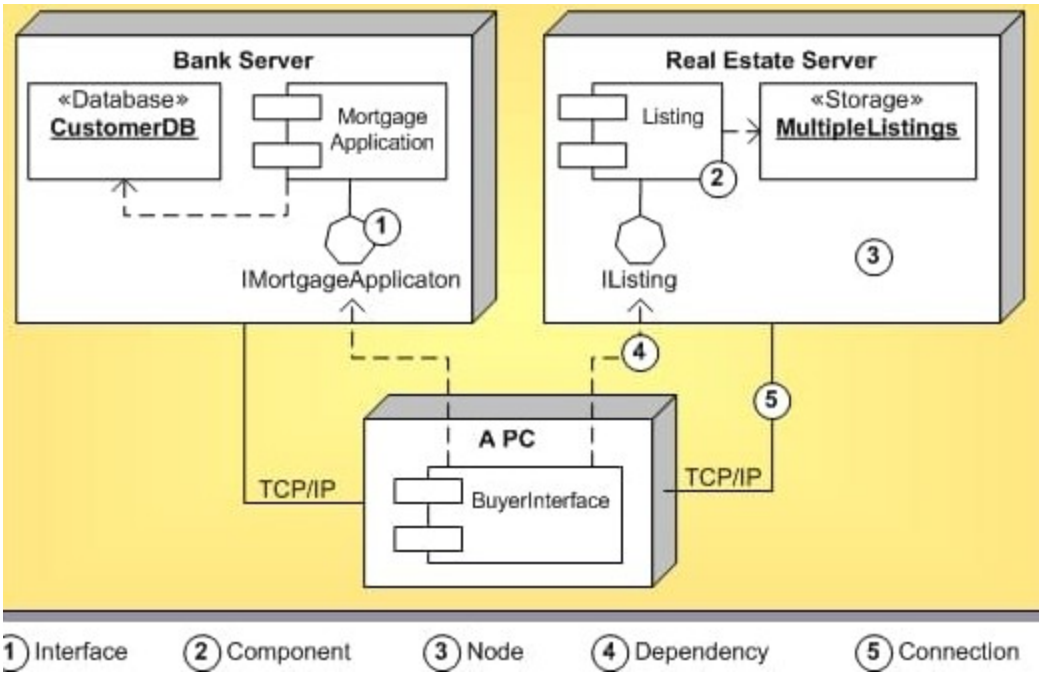
Des schémas d'architecture logicielle et de déploiement y figurent.

Par exemple :

le diagramme de composants dans la méthode UML



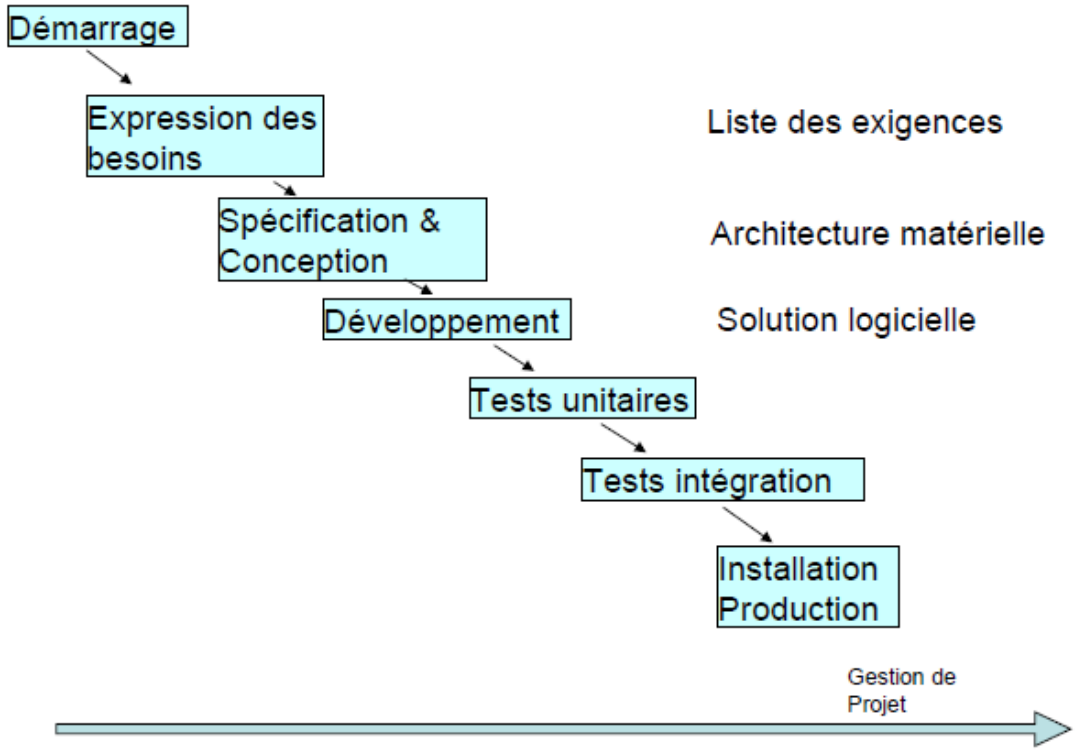
le diagramme de déploiement en UML



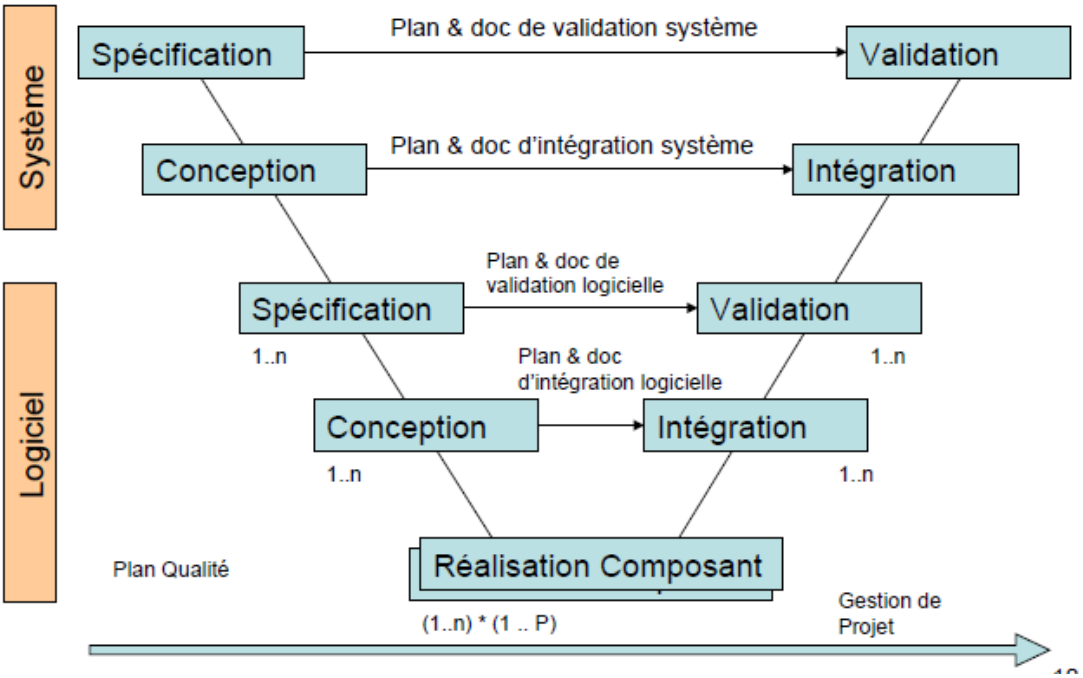
Section 4 Les types de tests

4.1 Les méthodes de gestion de projets – la place des tests

4.1.1 La méthode cascade



4.1.2 Cycle en V



Le logiciel doit résister aux erreurs de l'utilisateur et les tests suivants doivent être concluants :

- **Tests Unitaires** : menés par le développeur lui-même, ils consistent à vérifier la bonne exécution des fonctions dont il a la charge.
- **Tests d'Intégration** : menés par un testeur dédié, ils visent à tester la mise en commun de plusieurs composants et l'enchaînement de processus complets au-delà des simples fonctions unitaires.
- **Tests du système** complet, qui consistent à dérouler des scénarios complets, représentant les cas d'utilisation du logiciel, sans se préoccuper de l'implémentation ou des composants sous-jacents. Lors de l'évolution du système, on rejouera ces scénarios afin de valider la **non-régression**.

Le responsable des tests commence par déterminer l'objectif des tests qu'il va piloter, et rédiger les plans et document d'intégration et de validation.

Les objectifs peuvent être de :

- Trouver des défauts
- Acquérir de la confiance dans le produit,
- Acquérir des informations pour prendre une décision, lever des risques
- Ou prévenir des défauts

Quand l'objectif de test est précisé,

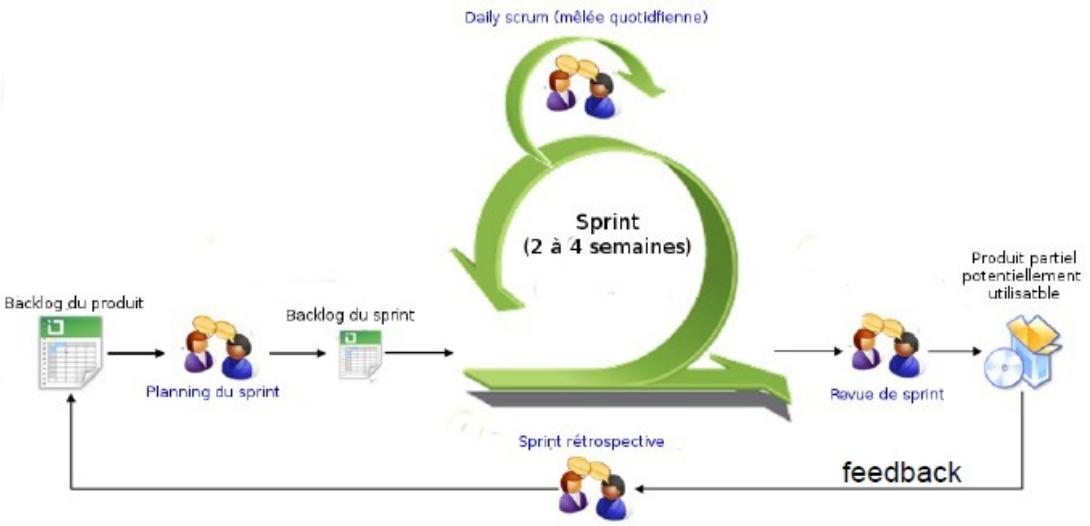
une stratégie doit être définie pour atteindre cet objectif et un planning doit organiser et distribuer l'effort de test dans le temps

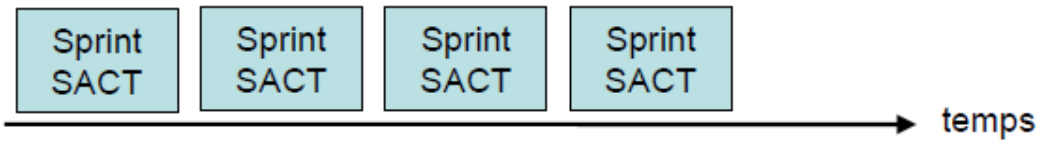
Quand cela est fait alors, le responsable des tests a un **Plan préparé que son équipe pourra suivre pour réaliser les tests**.

Généralement ce plan prend forme d'un document Word , plan d'intégration et plan de validation, qui définit par exemple :

- Le périmètre de test
- Les approches de test à adopter selon les niveaux de criticité des exigences
- Les niveaux de test
- Les critères d'entrée et les critères de sortie de chaque niveau de test
- les types de test
- Le planning des tests
- L'organisation du projet avec les rôles, les responsabilités, les réunions, leurs buts et leurs fréquences
- Les environnements de test, matériels, réseaux ...
- Et les outils de test

4.1.3 Agile Scrum





SACT signifie Spécification, Architecture (Conception) Codage, Test

4.2 Les types de test par indicateur qualité

Source https://www.softfluent.fr/blog/qualite-logicielle/

4.2.1 Capacité fonctionnelle

Évaluer le taux de couverture des fonctionnalités : le logiciel doit respecter les spécifications et surtout répondre aux attentes de l'utilisateur.

Vous pouvez créer **une matrice de respect des exigences**, lister les fonctionnalités et vérifier si elles sont implémentées et respectées avec des critères de type : aptitude, exactitude, interopérabilité, sécurité…

Pas d'outil automatisé, mais de la rigueur dans les documents de spécification fonctionnelle et de conception.

4.2.2 Fiabilité

Vérifier l'aptitude du logiciel à maintenir son niveau de service dans des conditions précises et pendant une période déterminée et notamment

- La fréquence des défaillances dues à des bugs

- L'aptitude à maintenir un niveau de service satisfaisant en cas de problèmes quel qu'il soit
- La capacité à rétablir son niveau de service et de restaurer les données directement affectées en cas de défaillance, en combien de temps ? avec quel effort ?

Le logiciel doit résister aux erreurs de l'utilisateur et les tests suivants doivent être concluants :

- **Tests Unitaires** : menés par le développeur lui-même, ils consistent à vérifier la bonne exécution des fonctions dont il a la charge. On privilégie les tests automatiques.
- **Tests d'Intégration** : menés par un testeur dédié, ils visent à tester la mise en commun de plusieurs composants et l'enchaînement de processus complets au-delà des simples fonctions unitaires. Quand cela est possible on privilégie l'intégration automatisée, dite intégration continue.
- **Tests du système** complet, qui consistent à dérouler des scénarios complets, représentant les cas d'utilisation du logiciel, sans se préoccuper de l'implémentation ou des composants sous-jacents. Lors de l'évolution du système, on rejouera ces scénarios afin de valider la **non-régression**.

4.2.3 Facilité d'utilisation

Prendre en compte les feedbacks des utilisateurs (formulaire de contact, téléphone, forum…) pour améliorer en continu l'expérience utilisateur

Effectuer des tests sur des groupes de personnes

- Les **tests d'acceptation**, ou recette, menés avec des utilisateurs pilotes, afin de valider l'adéquation aux besoins et la facilité d'adoption par ceux qui devront utiliser le logiciel régulièrement.
- Les **tests d'ergonomie.** Ces tests incluent tout autant la facilité d'utilisation que l'apparence visuelle

4.2.4 Performance

Vérifier le temps de démarrage : c'est une des clés voire la clé de succès de votre logiciel notamment pour les nouvelles générations habituées à obtenir tout de manière rapide et zappant sans état d'âme si l'efficacité n'est pas là.

S'assurer d'un temps d'exécution satisfaisant pour les tests suivants

- Les **tests de performance** consommation CPU, évolution de l'utilisation mémoire, nombre de requêtes par seconde, flux d'entrée/sortie… Le test de performance détermine la bonne exécution en mesurant les temps de réponse sans contexte particulier. Dans le cas d'un test de performance continu, il démarre dès le début des phases de développement, et est adapté à chaque étape du cycle de vie de l'application, jusqu'aux tests de charge complets.

- Les **tests de montée en charge** doivent être distingués des tests de performance et réalisés avec des outils permettant de simuler de nombreux clients simultanés. En augmentant le nombre d'utilisateurs par paliers, il détecte les éventuelles limites en capacité du système, pour valider la qualité de service avant déploiement

4.2.5 Maintenabilité

Les points à vérifier sont notamment

- **Faculté d'analyse** : facilité à diagnostiquer les déficiences, leurs causes ou à identifier les parties à faire évoluer
- **Faculté de modification** : facilité à modifier, remédier aux défauts ou à prendre en compte l'environnement
- **Stabilité** : facilité à anticiper les risques éventuels lors de modifications

4.3 Définition de la recette

Source : https://fr.wikipedia.org/wiki/Test_d%27acceptation

En informatique, le **test d'acceptation** (ou **recette**) est une phase de développement des projets, visant à assurer formellement que le produit est conforme aux spécifications (réponse donnée à un instant « t » aux attentes formulées). Elle s'inscrit dans les activités plus générales de qualification.

Cette étape implique, en la présence effective des différents acteurs du projet, maîtrise d'œuvre et maîtrise d'ouvrage, le déroulement rigoureux de procédures de tests préalablement décrits, et l'identification de tout écart fonctionnel ou technique.

Étapes :

La procédure de recette se déroule en deux étapes principales :

1. les tests système ;
2. les tests d'acceptation utilisateur.

Si la première étape a lieu chez le fournisseur, la deuxième se déroule en revanche généralement dans les locaux et avec les infrastructures du client.

Recette usine

La **recette usine** comprend tous les tests réalisés chez le fournisseur, avant la livraison. Elle comprend donc les tests unitaires, les tests d'intégration et les tests de validation.

D'un point de vue maîtrise d'ouvrage, la recette-usine correspond à une période de quelques jours pour valider que la livraison correspondra à sa commande.

En ce qui concerne les tests unitaires, d'intégration et de validation, ceux-ci sont considérés comme relevant des tests de qualification en amont, au niveau du fournisseur. Phase très importante pour celui-ci, car c'est elle qui lui permet de vérifier que son produit est de qualité et de faire accepter au client l'installation sur ses plateformes des tests et recettes pour les tests utilisateur métier VA (vérification d'aptitude), puis pour la VSR (Vérification de Service Régulier, voir ci-dessous).

Par extension, elle signifie aussi la période (généralement courte) durant laquelle le client procède lui-même à ses propres tests dans ses locaux, avant d'accepter les livrables.

À l'issue de la recette usine, le fournisseur et le client signent un procès-verbal de fin de recette usine, qui accompagne la livraison du produit et le cahier de recette.

Recette utilisateur

Lors de l'étape de **vérification d'aptitude** (VA) ou **vérification d'aptitude au bon fonctionnement** (VABF) (aptitude à répondre aux besoins exprimés dans le cahier des charges initial) ou **recette utilisateur**, le client réalise deux catégories de tests différentes.

D'un côté, une recette technique est effectuée afin de vérifier que le produit livré est techniquement conforme sur toute la chaîne de processus.

De l'autre, la maîtrise d'ouvrage contrôle l'aspect fonctionnel du produit lors de la recette fonctionnelle.

Recette fonctionnelle

La **recette fonctionnelle** a pour but la validation des fonctionnalités exprimées dans le cahier des charges et détaillées dans les spécifications fonctionnelles. La MOA procède donc à sa propre série de tests de validation.

Recette technique

Chargée de contrôler les caractéristiques techniques du produit livré, la **recette technique, ou VABE (vérification d'aptitude à la bonne exploitabilité)** regroupe les tests suivants :

- les tests d'exploitabilité : les tests de supervision, de sauvegarde... et en particulier les tests de respect des exigences d'architecture technique ;
- les tests de performance.

Si la VABF se déroule correctement et est validée, le client procède alors à la mise en service opérationnelle.

Une période de **vérification de service régulier** (VSR) commence donc par un premier déploiement sur un site pilote. Cette mise en production permet de valider le produit en conditions réelles.

À la différence des étapes précédentes, celle-ci se déroule pleinement en **environnement de production** avec des données réelles.

4.4 Documents livrables

Plusieurs documents accompagnent la procédure de recette :

4.4.1 Protocole de recette, ou stratégie de recette

Le protocole de recette est un document visant à clarifier intégralement la procédure de recette. Il précise scrupuleusement :

- les tâches du client ;
- les tâches du fournisseur ;
- la liste des documents à communiquer ;
- l'ordre des tests et le planning ;
- les seuils d'acceptation du produit.

4.4.2 Cahier de recette

Le cahier de recette est la liste exhaustive de tous les tests pratiqués par le fournisseur avant la livraison du produit.

La couverture des tests, en particulier ceux de non-régression lorsqu'il s'agit d'une nouvelle version d'un produit existant, pouvant être infinie, le cahier de recette doit préciser toutes les fiches de test passées par le fournisseur, ainsi que celles à passer dans l'environnement du client lors de la VABF.

4.4.3 Fiches de faits techniques

Les fiches de faits techniques visent à formaliser les écarts constatés en recette, et sont classifiés d'un commun accord entre fournisseur et client en : anomalies, et évolutions.

- Les anomalies, ou bugs, décrivent un écart du produit livré par rapport au comportement spécifié et attendu ; elles sont généralement issues d'une défaillance du fournisseur, et peuvent donner lieu à de futures corrections.
- Les évolutions correspondent à un écart du produit livré par rapport au comportement attendu ; elles sont généralement issues d'une défaillance ou lacune du client dans son expression de besoin, et peuvent donner lieu à des avenants ou un futur contrat.

4.4.4 Procès-verbaux

Pour clore chaque étape de la procédure de recette, un procès-verbal est rédigé. Celui-ci a pour objet de prononcer la réception et de mentionner les réserves émises par chacune des parties.

Toutes ces recettes peuvent être contractuelles et donner lieu à un procès verbal de recette qui permet de prononcer la réception, assortie ou non de réserves.

Ainsi le PV de recette usine conditionne le démarrage de la période de VA, celui de VABF celle de VSR.

Dans la marine, la recette d'un navire est l'acte qui permet le transfert de propriété à l'acquéreur si aucune réserve n'est émise.

Le recettage informatique se fait sur plusieurs dimensions : technique/fonctionnelle et usine/utilisateur.

En règle générale, la recette informatique s’organise autour de plusieurs parties prenantes :

- la MOA (maîtrise d’ouvrage) qui pilote le projet et dans le cas présent rédige le cahier des charges fonctionnelles, effectue la recette fonctionnelle, écrit les cas d’utilisation et les cas de tests fonctionnels.
- la MOE (maîtrise d’œuvre) qui conçoit la solution technique et assure la conformité technique (notamment tests d’intégration, tests unitaires, tests de charge)
- les testeurs qui exécutent les tests en s’appuyant sur un cahier de recette détaillé et un outil de bugtracking.

https://testacademy.fr/explique-moi-en-detail-le-processus-de-test/

Plan possible du cahier de test :

- **Introduction** : Vous allez y définir le contexte dans lequel va se faire cette recette
- **Généralités** : Vous allez ensuite expliquer les généralités, l'organisation nécessaire au bon fonctionnement des tests
- **Informations détaillées** : Puis, des informations détaillées sur le type de test à effectuer ainsi que les personnes qui y sont impliquées
- **Plan de tests** : Une section plan de test**s** vous permettra de présenter le format du plan et son utilisation
- **Exécution du plan de tests** : Ensuite, vous expliquez comment vont s'exécuter chaque partie des tests et comment remplir le plan avec les informations utiles.
- **Résultats** : ceux-ci seront alors résumés en fin de cahier
- **Approbations** : Lorsque les tests seront terminés, une section approbations vous permettra d'officialiser les résultats et leur adéquation avec le cahier des charges

Section 5 Étude de cas

5.1 Énonce

Le but est de rédiger un cahier de recette – acceptation client – dans le contexte d'un projet.

Le projet support pourrait être votre projet « la ruche connectée ».

Le document PlanDeTest.pdf propose 2 plans. Je propose de vous inspirer de l'exemple 1.

Le livrable est un document Word et éventuellement de feuilles Excel.