Daniel Muller

QUALIFICATION ET CYCLE DE VIE DU LOGICIEL

Guide d'introduction

Version V 0.*4*, *23/04/2010*
Etat : *En cours de rédaction*

SUIVI DES MODIFICATIONS

Version	Rédaction	Description	Date
0.1	Daniel Muller	- Conversion du document initial à la nouvelle charte graphique	19/02/2010
0.2	Daniel Muller	- Suppression du chapitre « Ingénierie logicielle » - Mise à jour du chapitre : Mise à jour du chapitre « Cycle de vie du logiciel »	25/02/2010
0.3	Daniel Muller	- Ajout du chapitre sur la stratégie de test	22/04/2010
0.4	Daniel Muller	- Ajout du chapitre sur les méthodes agiles	23/04/2010

Sommaire

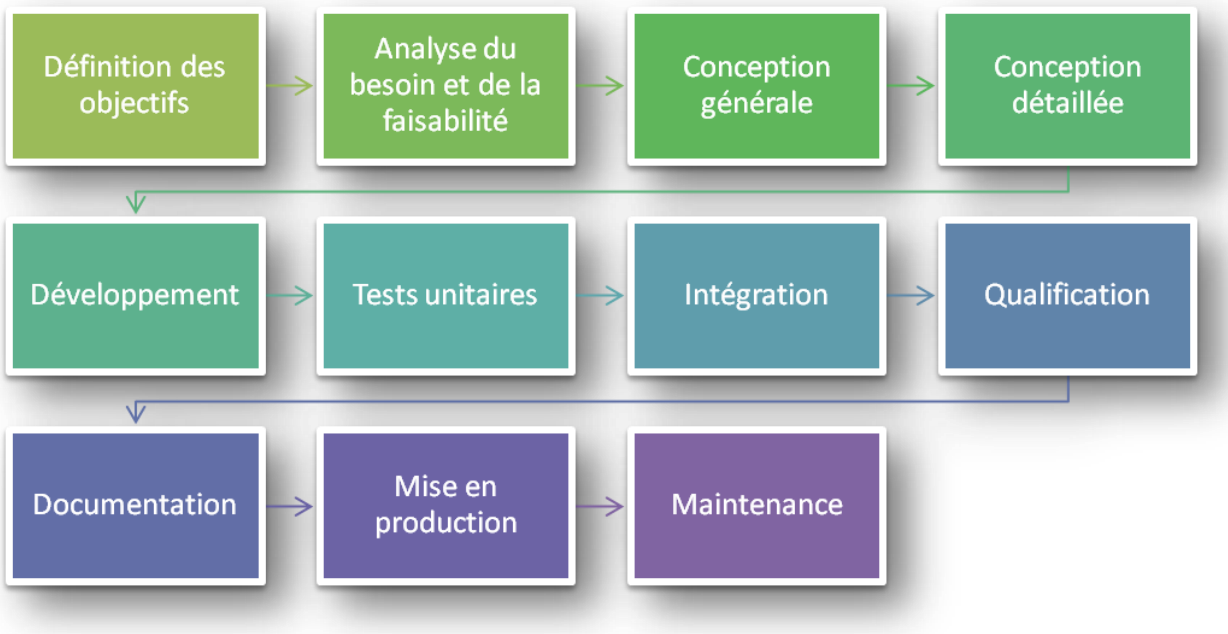
I CYCLE DE VIE DU LOGICIEL

La mise en production d'une application est l'aboutissement de nombreuses étapes où différentes équipes ont collaboré pour animer et consolider son développement. Ces étapes sont découpées selon un cycle déterminé avec pour objectif la validation du logiciel ainsi développé, c'est-à-dire la conformité de l'application avec les besoins exprimées.

Les erreurs subies durant un développement peuvent engendrer des coûts élevés lorsqu'il faut effectuer les corrections nécessaires. Plus cette erreur intervient tardivement dans le développement, plus il est difficile de la rattraper. En adoptant un modèle de cycle de vie, il est possible de détecter ces erreurs au plus tôt et ainsi de les rectifier en garantissant de ce fait une maitrise de la qualité du logiciel livré, un respect des délais et des coûts impartis.

1. Cycle de vie générique

Un projet informatique va généralement suivre le cycle de vie suivant :



- La **définition des objectifs** offre un cadre au projet et s'inscrit dans une stratégie globale de l'entreprise.

- L'**analyse du besoin et de la faisabilité** décrit le moment où les souhaits du client sont formalisés dans un recueil d'expression de besoin. Les contraintes techniques, fonctionnelle et environnementales sont alors mises en avant afin de vérifier si l'application est réalisable ou non. Les personnes à l'initiative du projet se réunissent, consolident leur besoin et rédigent les documents suffisamment explicites pour être compris par les équipes qui vont prendre la suite, soit la MOA et la MOE).
  - Etude d'avant projet
  - Rédaction de l'expression de besoin
  - Rédaction du document de conception générale

- La **conception générale** définit les spécifications de l'architecture générale du logiciel à développer.

- La **conception détaillée** définit clairement chaque élément constituant le logiciel qu'il soit technique ou fonctionnel.

- Le **développement** permet au logiciel de prendre corps et traduit à travers différentes interfaces les besoins exprimés et respecte les spécifications techniques détaillées.

  L'ingénierie logicielle s'intéresse particulièrement à cette phase. Plusieurs modèles peuvent être utilisés pour suivre précisément chaque étape pour fournir au final un produit de qualité. Le plus courant est le modèle en cascade où les étapes du projet se succèdent chronologiquement, chacune se terminant par une phase de vérification.

- Les **tests unitaires** vérifient chacun des organes composant le logiciel et leur conformité par rapport aux spécifications.

- La **phase d'intégration** permet de vérifier que l'interfaçage entre les différents modules du logiciel est cohérent. Le bilan de ces tests est consigné dans un document dédié.

- La **phase de qualification** (plus communément appelée recette) vise à vérifier la conformité de l'ensemble du logiciel par rapport aux exigences.

  Le chef de projet qualification et son équipe prennent alors la main pour vérifier l'adéquation du logiciel développé par rapport aux exigences. Leur intervention est cruciale car, de leur faculté à maitriser l'applicatif et à comprendre les documents, va dépendre de la qualité du logiciel livré au final aux utilisateurs.

  Un projet conséquent va passer par les étapes de qualification suivantes :
  - Pré-recette
  - Recette technique et fonctionnelle
  - Certification
  - Pré-recette
  - Recette technique et fonctionnelle
  - Certification

- La **documentation** réunit les informations nécessaires pour l'exploitation du logiciel en production, un manuel utilisateur et tout autre écrit permettant son amélioration.

- La **mise en production** consiste à fournir un package et une documentation détaillé permet au client d'installer simplement le logiciel sur l'environnement final. Lors de cette phase, il est également important de vérifier le comporter du logiciel ainsi développé face aux autres applications environnantes et s'assurer qu'il n'existe aucune interférence.

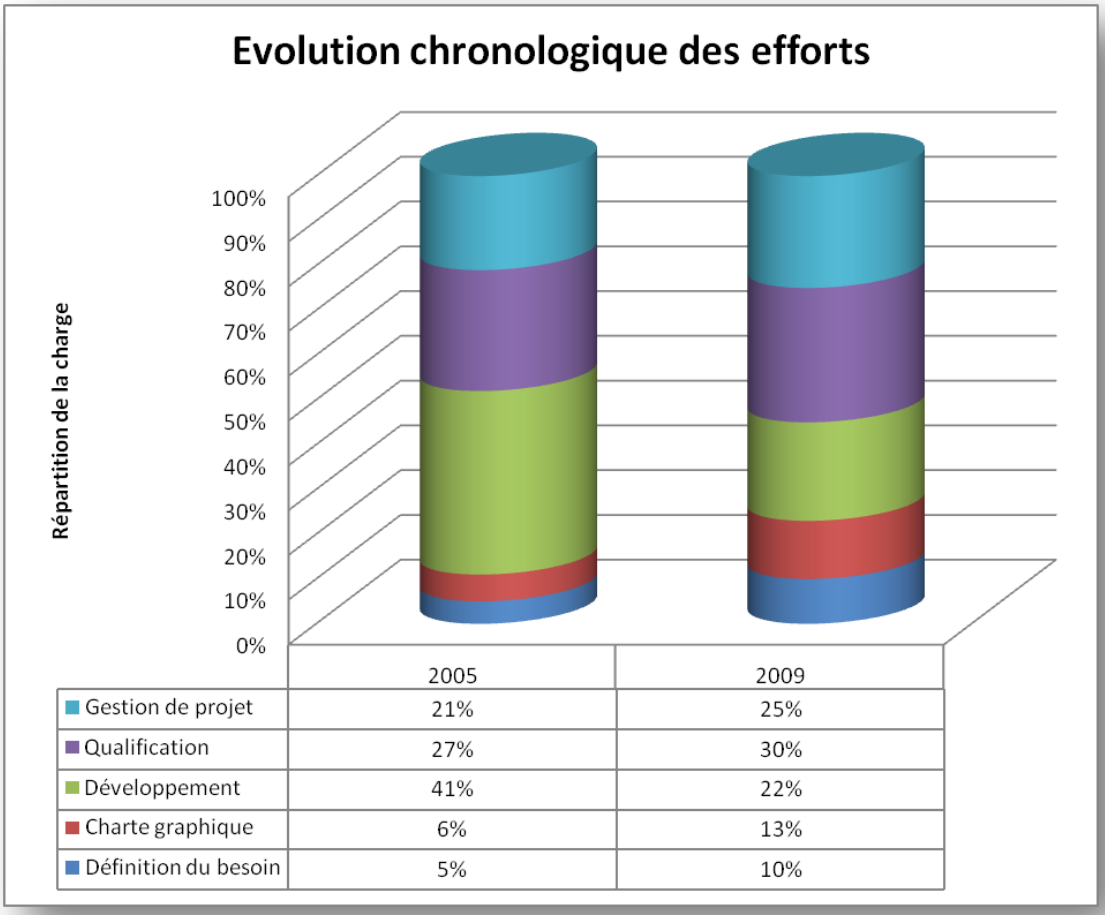
- La **maintenance** est une période définie par contrat où le logiciel subit des maintenances correctives en cas de panne ou des phases de maintenances préventives définie par avance.

Pour mettre en pratique les différentes phases du cycle de vie du logiciel, deux modèles sont généralement utilisés :

- Le modèle en cascade.
- Le modèle de cycle en V

2. Répartition de l'effort

Depuis quelques années, les responsables de projets ont pris conscience de l'intérêt des tests et l'impacté de manière sensible dans le cycle de conception de leur produits.



	2005	2009
Gestion de projet	21%	25%
Qualification	27%	30%
Développement	41%	22%
Charte graphique	6%	13%
Définition du besoin	5%	10%

Que peut-on conclure à la lecture de ce graphique ?

- Un effort est consenti quant à une meilleure définition du projet. La prise en comptes des exigences et le concours de toutes les parties prenantes dans cette phase initiale est devenu un élément important du modèle CMMI.

- Avec des interfaces de plus en plus riches et complexes, le temps alloué aux éléments visuels du logiciel a doublé depuis 2005. Il n'est pas rare aujourd'hui que des informaticiens spécialisés se chargent uniquement de cet aspect, le design étant devenu un métier à part entière avec ses propres outils : les technologies Adobe Flex, Adobe Designer ou Microsoft Silverlight en sont des exemples.

- Cela peut paraître étonnant à première vue mais la phase de développement a été réduite de moitié en 4 ans. On peut expliquer cela par la dissociation du design et des composants logiciels en eux-mêmes, mais également par l'utilisation progressive d'éléments développés dans les projets précédents comme le conseille l'un des paliers CMMI. Les logiciels de développement ont particulièrement évolué et apporte une aide conséquente au développeur, à l'instar de Microsft Visual Studio. A noter que cette phase intègre également les commentaires insérés dans le code, les documents annexes ainsi que les tests unitaires.

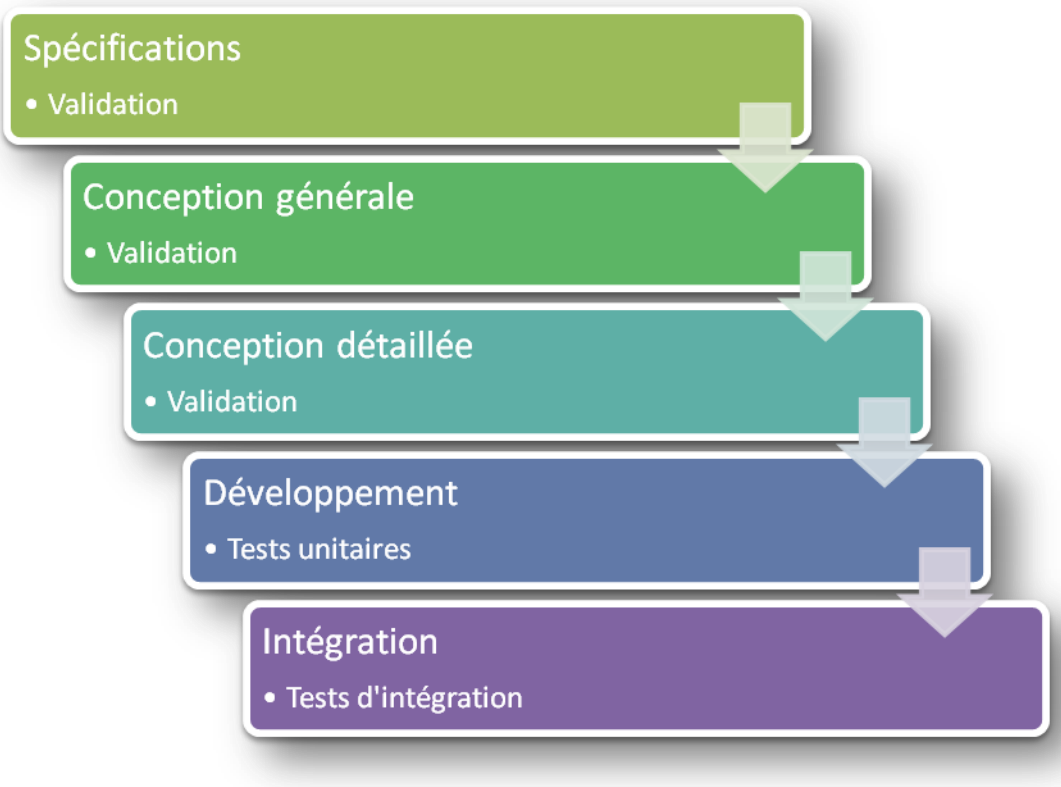
- La phase de qualification prend une place prépondérante dans le projet, pas moins d'un tiers du temps lui est désormais consacré. On y inclut la phase de conception, d'exécution des tests, la résolution des anomalies et également les tests d'intégrations effectués au préalable.

- Le temps consacré à la gestion de projet est conséquent, surtout depuis l'adoption de normes telles que CMMI : des réunions avec les différents acteurs sont plus nombreuses, le suivi plus directif et la production de documents également plus importante.

3. Modèle en cascade

Ce modèle est issu de l'expérience des ingénieurs du bâtiment et a connu sa formalisation dans les années 1970. Il part du principe que l'on ne peut ériger les murs d'une maison seulement si les fondations sont solides. Egalement, la toiture ne peut être posée que lorsque les murs sont correctement posés et capables de supporter le poids inhérent à la structure supérieure. Plus tard est détecté la défaillance, plus graves peuvent être les conséquences.

Appliqué à l'informatique et au développement du logiciel, le modèle en cascade propose une étape de vérification à l'issue de chaque phase du cycle. Ainsi, il est possible de détecter les anomalies survenant à chacune de ces phases et de les corriger si nécessaire avant de passer à la phase suivante.



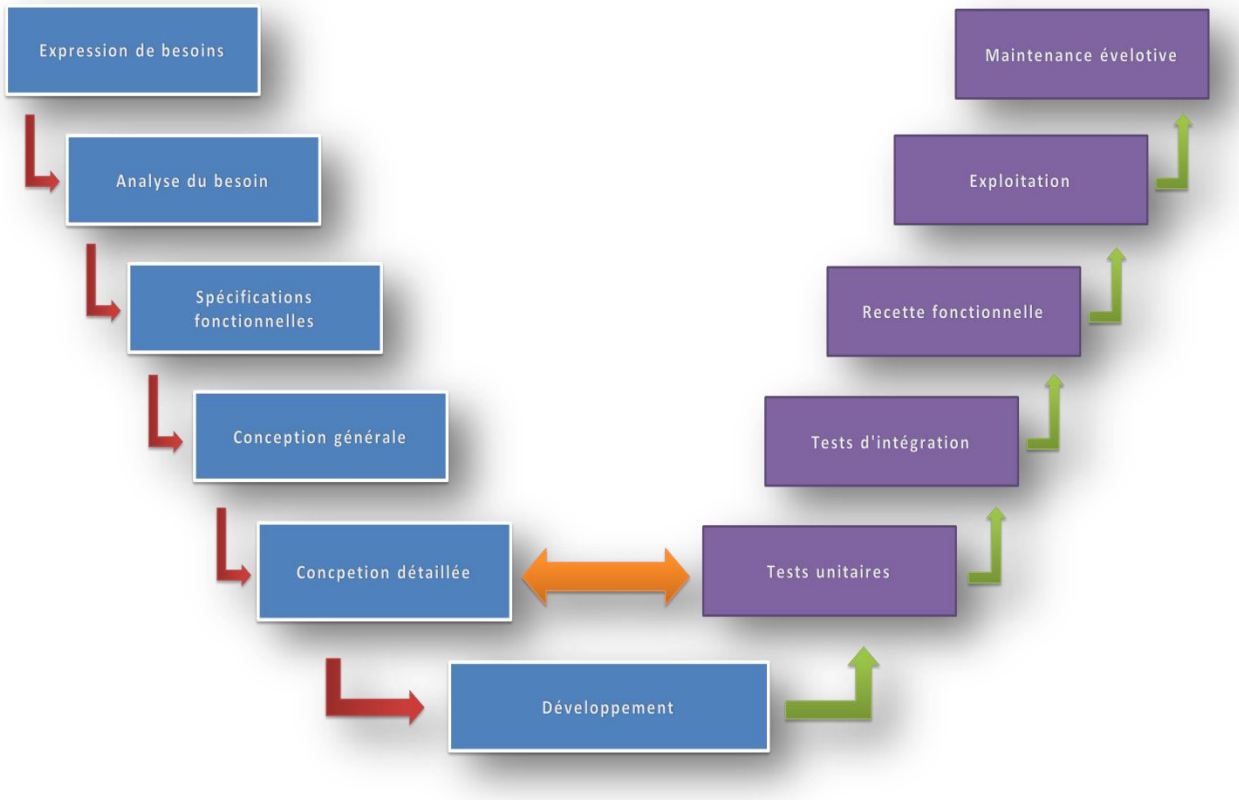
4. Modèle de cycle en V

Si le modèle de cycle en cascade est assez simple à mettre en place, il peut cependant manquer de réactivité lorsqu'il s'agit de l'appliquer à un projet de développement conséquent. Autre inconvénient, les équipes chargées de la définition et de la conception du logiciel n'interviennent qu'eu début du projet. En cas d'anomalie d'ordre fonctionnelle, il est alors difficile de les réintégrer dans le cycle pour apporter les solutions nécessaires.

C'est dans ce contexte qu'à été conçu le modèle de cycle en V. Il oblige les équipes MOA, MOE et Qualification à s'impliquer fortement tout au long du projet.

Ainsi, dès le lancement de la phase de conception, l'équipe de recette est mise à contribution pour la rédaction des scénarios de tests de qualification.

Cette approche permet de réunir l'ensemble des intervenants au projet et de mener ensemble une analyse plus approfondie.



5. Méthode agile

Cette méthode s'applique lorsque les problématiques suivantes sont rencontrées :

- L'application fait l'objet d'évolutions régulières avec des cycles de développement courts.
- Les spécifications fournies sont volatiles ou peuvent subir des changements de dernière minute.

Dans ce contexte, appliquer l'une des méthodologies telles que le modèle en cascade ou le modèle de cycle en V pour concevoir un tel produit se révèle inadapté, car trop énergivore en temps et en ressources.

En pratique, les équipes de conceptions et qualité en viendraient à subir le projet sans aucune anticipation possible.

L'objectif de la méthode agile, édicté en 2001, est de satisfaire un niveau de qualité suffisant et de rendre la tâche de développement la plus souple possible.

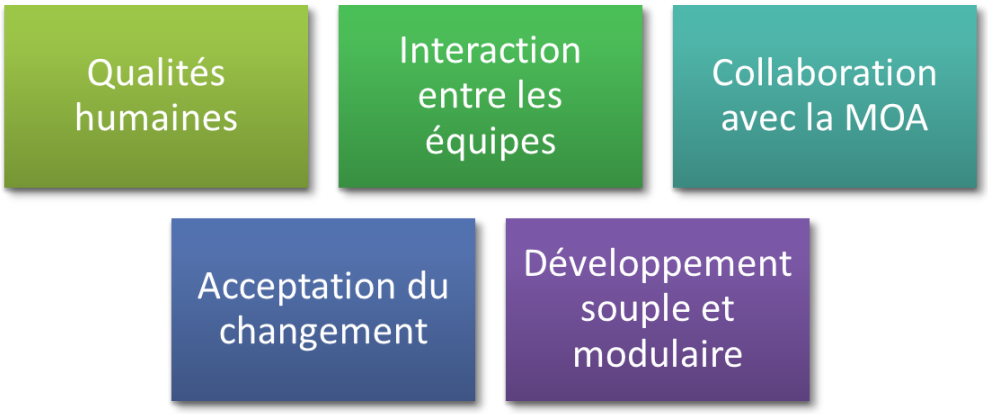
La méthode agile se caractérise par :

- Une approche adaptative où les différentes équipes participant au projet (MOE, MOA, Qualité) doivent faire preuve de souplesse pour se plier à des changements subits.
- Une importance donnée aux qualités humaines plutôt qu'à des process standardisés. Chaque membre du projet est clairement identifié, ses compétences sont exploitées au mieux. Revers de la médaille, l'appel à la responsabilité est primordial et il faut tenir compte des défauts inhérents à chaque personne impliquée.

Le manifeste des fondateurs de la méthode agile préconise ceci :

« La méthode agile apporte une meilleure façon de développer des logiciels en le faisant et en aidant les autres à le faire. »

Les éléments ci-dessous sont ainsi les piliers de cette méthodologie :



La priorité est donnée :

- Aux personnes, à leurs compétences et à leur interaction ;
- A des documentations fonctionnelles succinctes mais à jour avec les évolutions du projet ;
- A un développement respectant des normes strictes, à un code commenté, à des modules réutilisables ;*
- A une collaboration constante avec la MOA émettrice du besoin, à une réelle compréhension de ses besoins de manière à fournir une solution conforme à ses attentes.
- A une acceptation du changement, à la souplesse accordée à un planning qui s'adapte en fonction des priorités de la MOA.

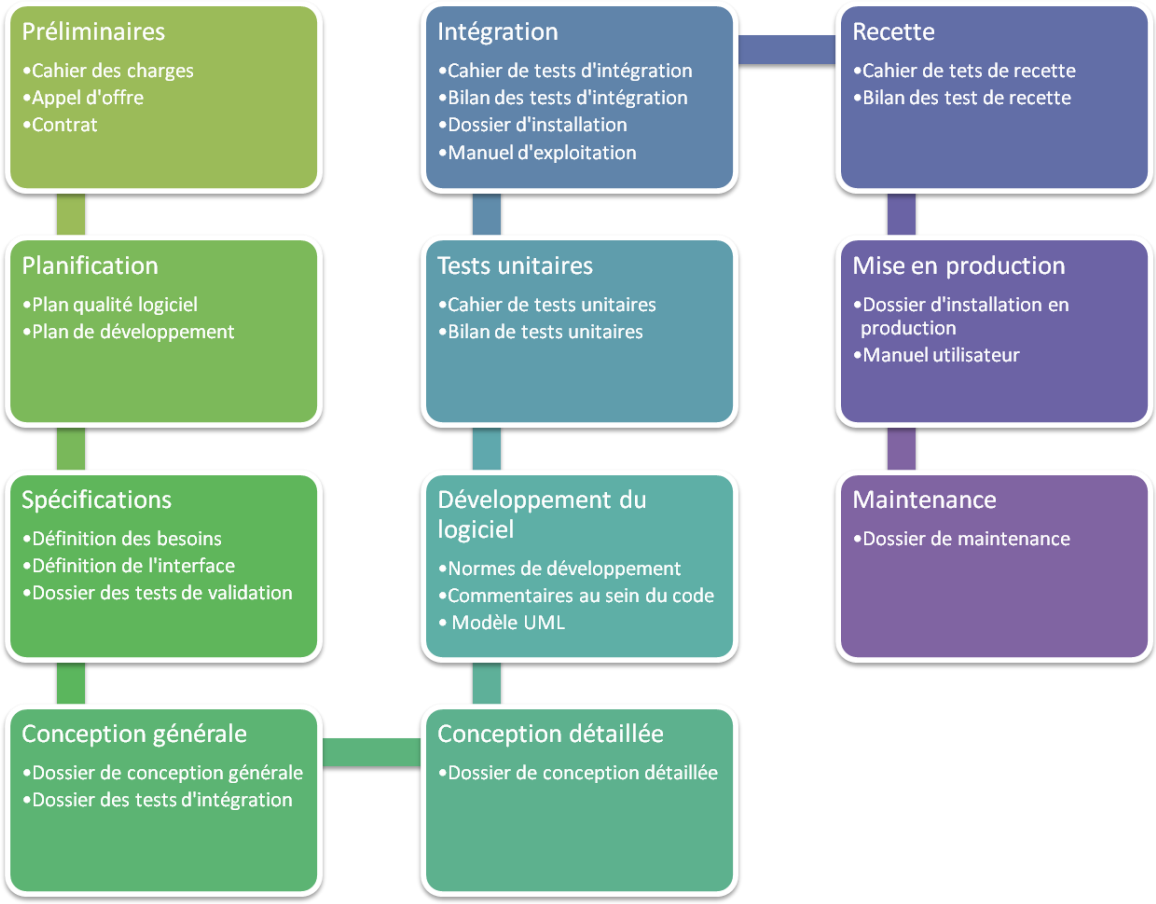
En pratique, la méthode agile doit respecter les principes suivants :

- Livrer régulièrement des versions fonctionnelles de l'application. La MOA pourra alors choisir une mise en production de la version actuelle ou proposer des modifications et attendre la prochaine livraison. L'intervalle entre chaque livraison peut varier de deux semaines à deux mois selon les évolutions attendues.
- Accepter, voire anticiper les requêtes de la MOA, même en cours de développement. Il peut arriver que celle-ci demande l'ajout d'une nouvelle fonctionnalité ou, à contrario, de supprimer un module déjà développé.
- L'aspect humain étant important, le projet doit se construire autour de personnes compétentes, motivées et responsables. Charge à la hiérarchie de leur fournir tous les éléments nécessaires pour un travail optimal.
- Faciliter la communication, l'idéal étant que les membres cohabiter au sein d'un même environnement.
- Le rythme de travail doit être soutenable de manière à ne pas démotiver les équipes et maintenir un planning prévisible.

- Le développement doit être réalisé de manière professionnelle : respect des normes, un code commenté et aéré, une conception modulaire.
- Fournir la réponse la plus simple possible à la MOA afin que celle-ci puisse être réutilisée ultérieurement.
- Régulièrement, les parties prenantes doivent échanger sur les difficultés rencontrées et apporter des solutions pour travailler plus efficacement.

6. Sources documentaires du projet

A l'image du cycle de vie du logiciel, les normes qualité en place impliquent la rédaction de documents qui vont illustrer le développement de l'application.



Les deux tableaux qui suivent montrent, en entrée, les événements déclencheurs et en sortie les documents générés :

- Le premier tableau est consacré à l'initiation du projet, le second au développement et au test de l'applicatif.

- Pour chacun des tableaux, la première ligne définit l'objectif à atteindre, la seconde les processus en entrée, la troisième les processus en sortie.

Avant-projet :

Phase de Qualification :

Etape 1

- **Objectif** : Préparation de la qualification
- **Entrée** :
  - Conception fonctionnelle
  - Conception technique
  - Recueil des exigences
- **Sortie** :
  - Exigences de tests
  - Plan de test

Etape 2

- **Objectif** : Organisation de la qualification
- **Entrée** : Plan de test
- **Sortie** :
  - Scénarios de test
  - Campagnes

Etape 3

- **Objectif** : Qualification de l'application
- **Entrée** : Exécution des scénarios de recette
- **Sortie** : Couverture complète des scénarios

Etape 4

- **Objectif** : Gestion des anomalies
- **Entrée** : Anomalies ouvertes
- **Sortie** :
  - Anomalies corrigées
  - Anomalies closes
  - Fiches d'anomalies avec description de la correction

Etape 5

- **Objectif** : Clôture de la qualification
- **Entrée** :
  - Fiche de changement
- **Sortie** : Conception et exécution des cas de tests issus des fiches de changement

Les documents les plus importants pour le chef de projet Qualification sont les suivants :

- Le dossier des spécifications détaillées fonctionnelles
- Le dossier des spécifications détaillées techniques

7. Spécifications fonctionnelles

L'objectif de ce document est de décrire de manière la plus précise possible les différentes fonctionnalités que compte offrir l'application sans aborder les problématiques techniques.

Dans l'idéal, la conception fonctionnelle fournit les informations suivantes :

- Une description détaillée des fonctionnalités de l'applicatif
- Le modèle conceptuel des données circulant dans l'application
- Une maquette et un schéma représentant la cinématique de l'application
- La sécurité intrinsèque
- Les profils des utilisateurs ciblés par cette application

8. Spécifications techniques

Ce document présente l'application sous un aspect purement technique. Il dresse la liste des technologies utilisées et décrit sous un angle « Développeur » le fonctionnement de chaque module et leur interaction.

On doit pouvoir trouver les informations suivantes :

- Le type d'application : Web, exécutable, client / serveur …
- La technologie utilisée pour son développement : .Net, Java, VB …
- L'environnement logiciel : version du Framework, du navigateur s'il s'agit d'une application Web, du système d'exploitation …
- L'environnement matériel tant pour le poste client que pour le serveur.
- Si l'application fait appel à une base de données :
    - Le nom du SGBD,
    - Le schéma relationnel
- La liste et une brève description des tables
- Une description technique de chacune des fonctionnalités de l'application.

Il arrive parfois qu'un seul document soit rédigé, résumant ainsi les deux aspects techniques et fonctionnels de l'application. On croit ainsi gagner du temps mais l'expérience montre que ce « raccourci » est particulièrement dangereux.

Ainsi, il est arrivé qu'au moment du pilote, on se rende compte que le document ne donnait aucune spécification quant à l'environnement logiciel. Une fois l'application testée en environnement de production, de graves dysfonctionnements ont surgi du fait de la version du Framework et du navigateur. Conséquence : résoudre ce problème a pris beaucoup de temps que ne l'aurait exigé une rédaction consciencieuses des documents.

Il est donc important de dissocier les deux facettes du projet afin de capitaliser au mieux les connaissances des équipes rédactrices. Plus ces documents seront de qualités, plus précises seront les exigences de tests rédigées par le chef de projet Recette. Son équipe sera alors à même de traquer l'intégralité du projet et proposer au final un livrable conforme aux attendus.

A partir de ces documents, le chef de projet va pouvoir construire un plan de test cohérent et assurer que l'intégralité de l'application aura été analysée par son équipe.

Avant de concevoir les scénarios, il réunit dès l'avant-projet tous les acteurs pour collecter les attentes de chacun et rédige un recueil des exigences cohérent et complet. Sont impliqués notamment :

- Les acteurs métier,
- Les évaluateurs de l'application une fois celle-ci déployée,
- Les responsables sécurité si l'application s'intègre dans un domaine existant,
- Les responsables des équipes de développement,
- Des experts techniques ou fonctionnels qui peuvent intervenir ponctuellement sur des parties précises de l'application.

Une fois cette collecte terminée, les exigences vont être regroupées en deux catégories :

- Les exigences fonctionnelles,
- Les exigences non fonctionnelles.

9. Exigences fonctionnelles

Une exigence doit concerner un besoin élémentaire fourni par l'application ou un acte précis sur une entité métier identifiée.

Exemples :

- L'application est disponible en deux langues : français, anglais,
- L'utilisateur peut accéder aux informations postales du client,
- L'utilisateur a accès à une nouvelle fonctionnalité : vente à crédit.

10. Exigences non fonctionnelles

Une telle exigence porte sur un point technique particulier où sur un élément lié à l'environnement du client. Généralement, ces exigences concernent :

- Les besoins techniques (volumétrie, connexions simultanées, temps de réponse ...).
- Les questions de sécurité.
- Les contraintes techniques liés à l'environnement du client où des problématiques de développement.
- Les exigences du client impactant l'application mais ne provenant pas du métier lui-même.

Exemples :

- L'application doit être disponible de 7h à 21h,
- 500 utilisateurs doivent pouvoir accéder à l'application en simultanée,
- L'application doit être compatible avec l'architecture logicielle existante.
- L'utilisateur doit mettre moins de 15 minutes pour parcourir l'ensemble de l'application.

Tout au long du projet, le Chef de Projet Recette va maintenir des engagements particuliers grâce à la gestion des exigences. Notamment, il est chargé de :

- D'identifier clairement quels sont les objectifs que l'application doit atteindre,
- D'établir, puis de mettre à jour les liens entre les exigences et les travaux réalisés,
- De vérifier la cohérence des documents nécessaires à la qualification de l'application.

II DEFINITION DU PROJET

Cette phase initiale a pour objectif de rechercher, puis fournir les équipes qui interviendront tout au long du projet.

Les principaux acteurs sont :

- La maitrise d'ouvrage définit le projet, prend en charge le financement et est conseil pour les aspects métier.
- La maitrise d'œuvre est responsable du développement et gère également des éléments techniques du projet.
- La qualification joue les intermédiaires entre maitrise d'œuvre et maitrise d'ouvrage afin de valider le logiciel tant au niveau fonctionnel qu'au niveau technique.
- La conduite du changement réceptionne le logiciel terminé, assure la transition avec les équipes de production et les utilisateurs finaux.

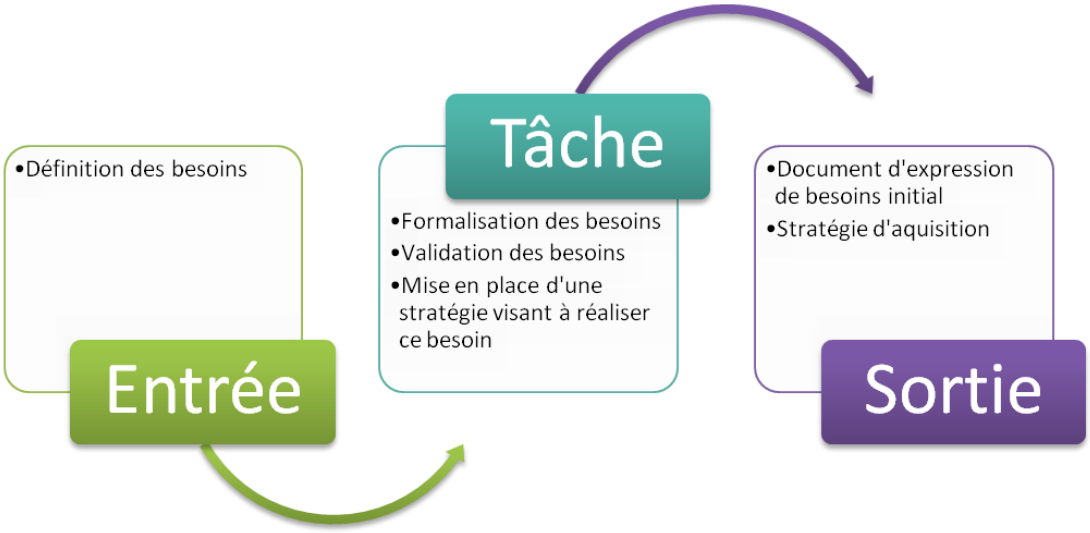
Cette phase met en place la relation client/fournisseur à travers les processus suivants :

- Définition et acquisition du produit
- Propositions contractuelles

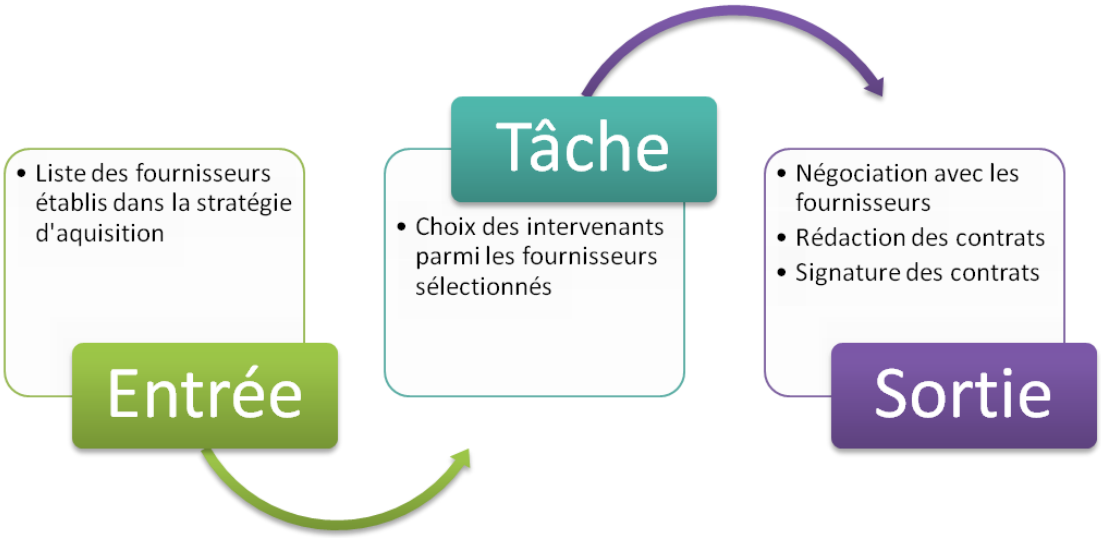
1. Définition et évaluation du produit

Cette phase a pour but de formaliser le besoin et de sélectionner le personnel qui prendra part au projet. Elle suit les processus suivants :

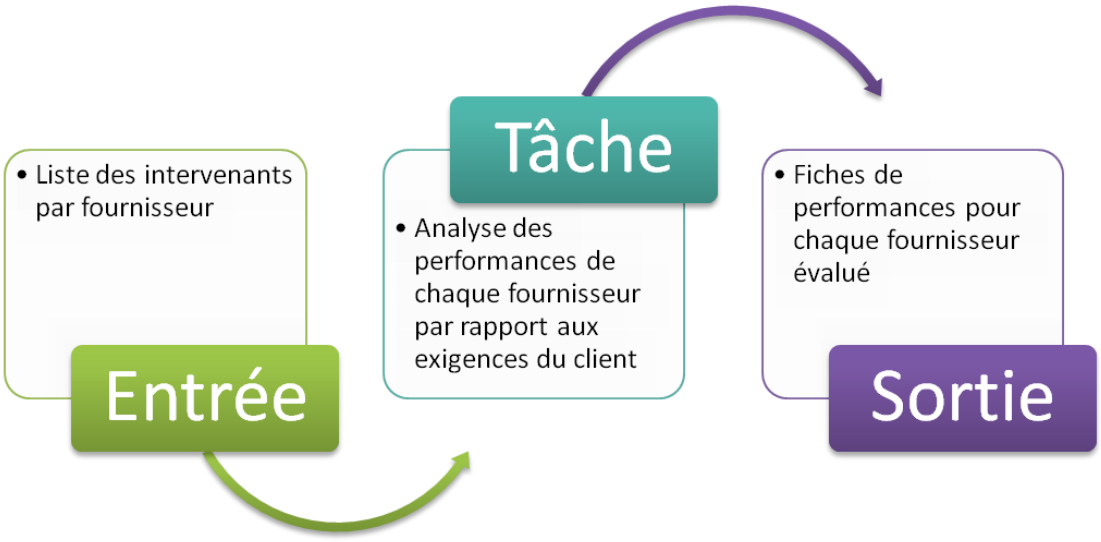
- La description des besoins



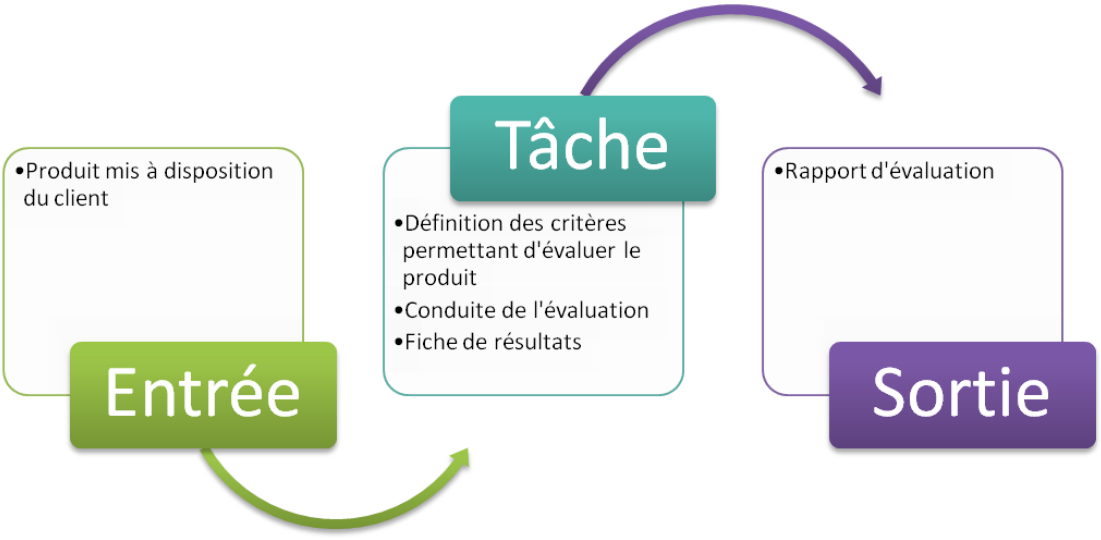
- La sélection des intervenants :



- La gestion des fournisseurs :



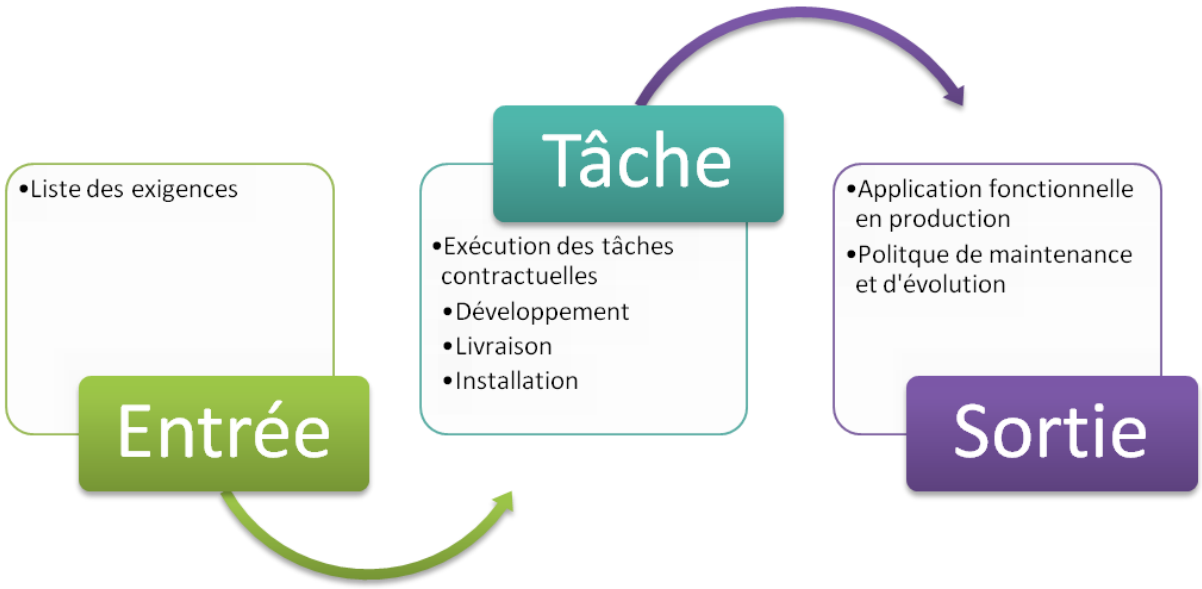
- L'évaluation du produit



2. Proposition contractuelle

Cette phase a pour but de formaliser :

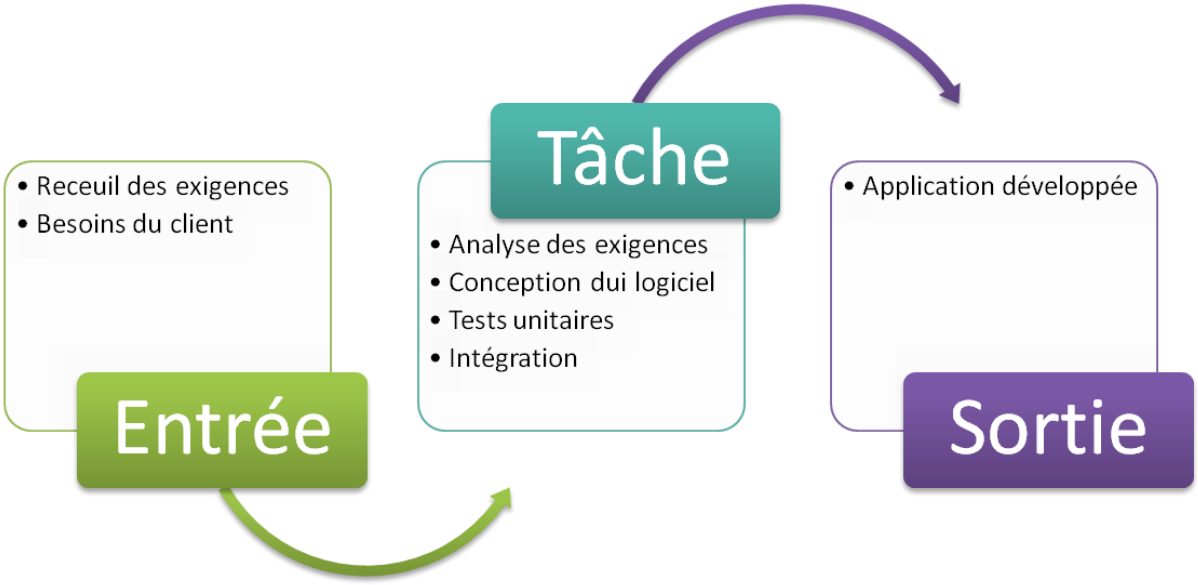
- La livraison du produit :



III ORGANISATION ET PLANIFICATION DU PROJET

Cette phase correspond au cœur du projet puisqu'il s'agit de donner une réalité aux souhaits du client.

Le processus générique est le suivant :



1. Note de lancement

Une fois le budget au projet alloué, la maitrise d'ouvrage édite une note de lancement, point de départ officiel du projet. Cette note décrit succinctement le fonctionnement de l'application à développer et fixe les objectifs à atteindre. Surtout, elle indique les intervenants choisis pour diriger le projet et les moyens qui leur sont alloués.

2. Plan assurance qualité

Le service qualité rédige de son côté un plan d'assurance qualité. Ce document dresse la liste des actions qualité qui seront menées et décrit le système de management qualité adapté au projet.

Trois types d'actions garantissent l'assurance qualité du projet :

- **La revue :**

La cohésion du projet est analysée et sa pertinence par rapport aux besoins du client est vérifiée.

- **L'inspection** :

Des points de contrôle sont disposés durant tout la période de développement. A chacun de ces points, une inspection qualité est entreprise.

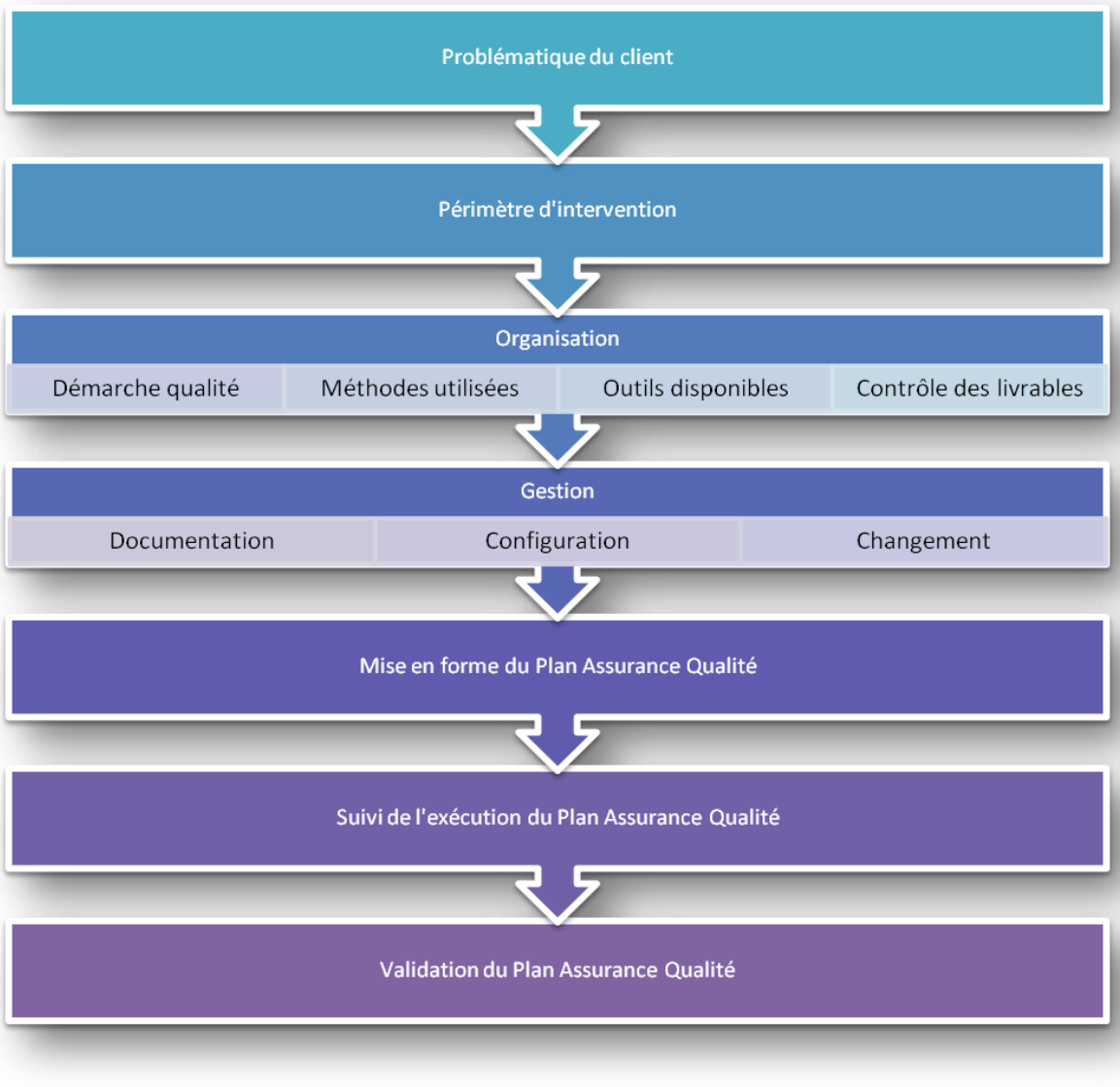
- **L'audit** :

Une fois le produit développé, on pratique un audit pour analyser les écarts et surtout expliquer les problèmes rencontrés durant la période de codage.

Le système management qualité, quant à lui, doit expliciter les informations suivantes :

- Quelle est la démarche qualité utilisée ?
- Quels sont les outils mis en place ?
- Quels sont les contrôles planifiés ?

Le schéma ci-dessous définit comment le plan d'assurance qualité d'un projet est conçu :



Le plan d'assurance qualité est sous la responsabilité du chef de projet désigné dans la note de lancement.

3. Plan de développement

Ce document détaille comment le chef de projet Etudes va conduire le développement de l'application.

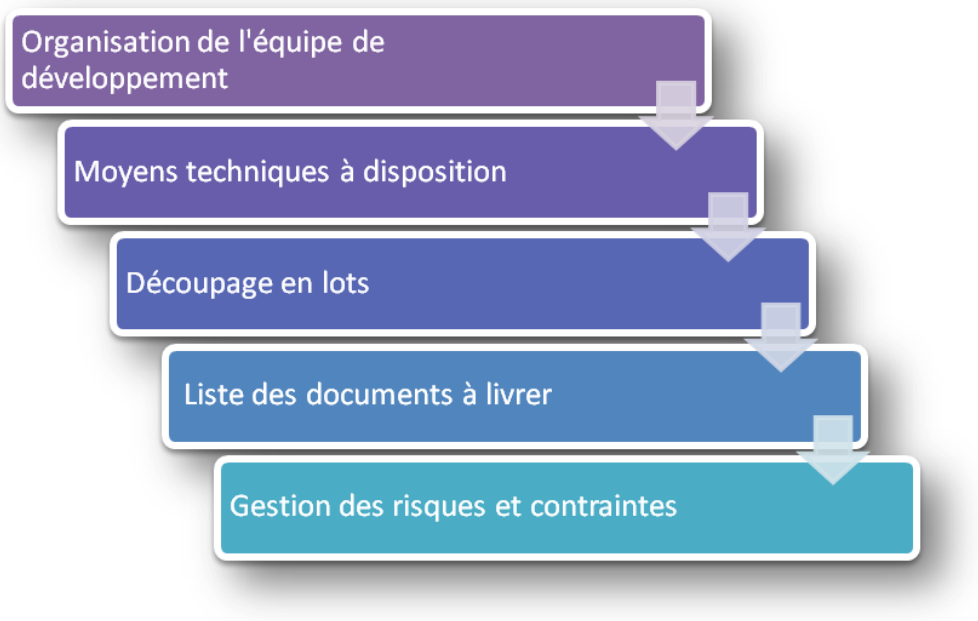
Il décrit principalement :

- Les moyens mis à sa disposition en termes humains et techniques,
- Le découpage du développement en lots avec les objectifs à atteindre par chacun d'eux,
- Le planning prévisionnel,
- Les documents à rédiger,
- Les méthodes utilisées pour gérer le projet.

Le plan de développement doit également préciser comment les différentes contraintes ont été intégrées au projet :

- Respect du planning et du budget alloué,
- Contraintes techniques,
- Contraintes fonctionnelles,
- Respect des normes qualité.

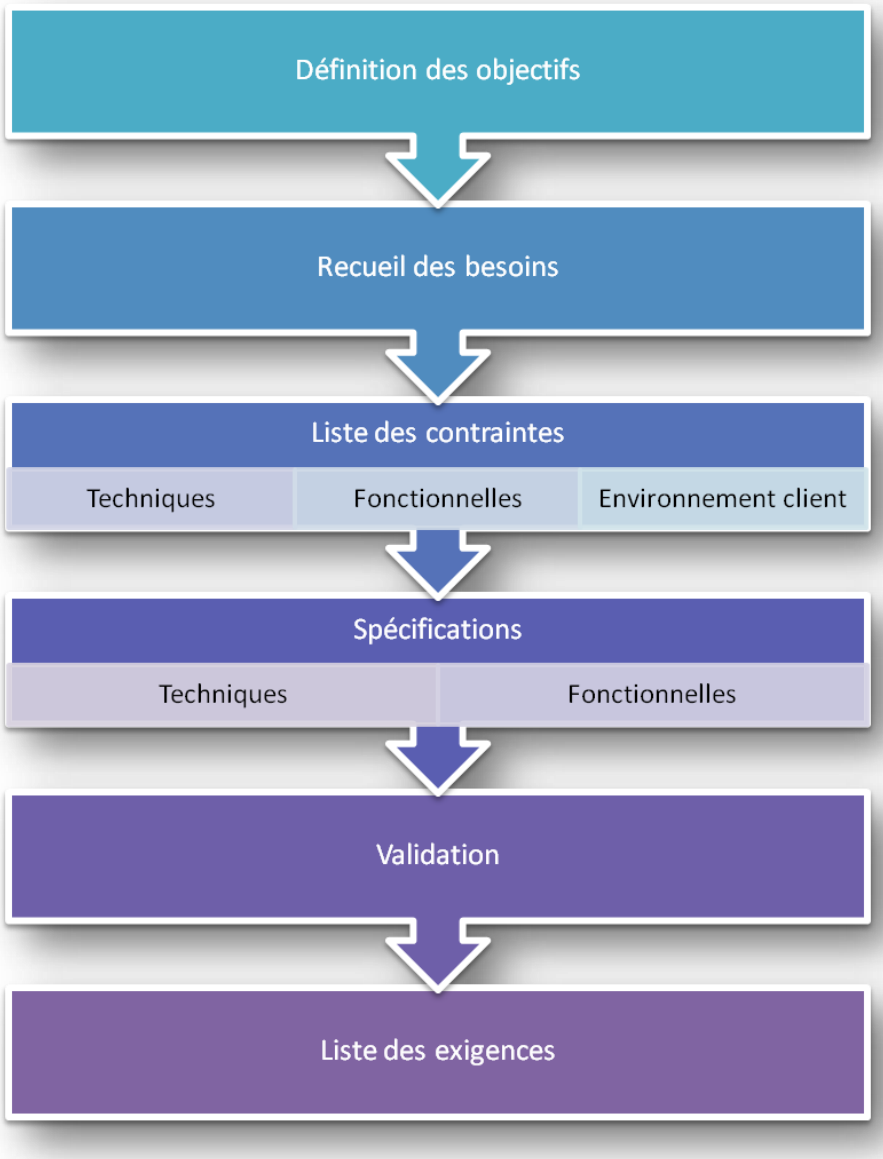
Le schéma ci-dessous montre le processus de création du plan de développement :



IV DEFINITION DES SPECIFICATIONS

Les spécifications permettent de dresser les besoins tant techniques que fonctionnel pour, au final, fournir une liste d'exigences exhaustives.

Ces exigences sont l'aboutissement des actions suivantes :

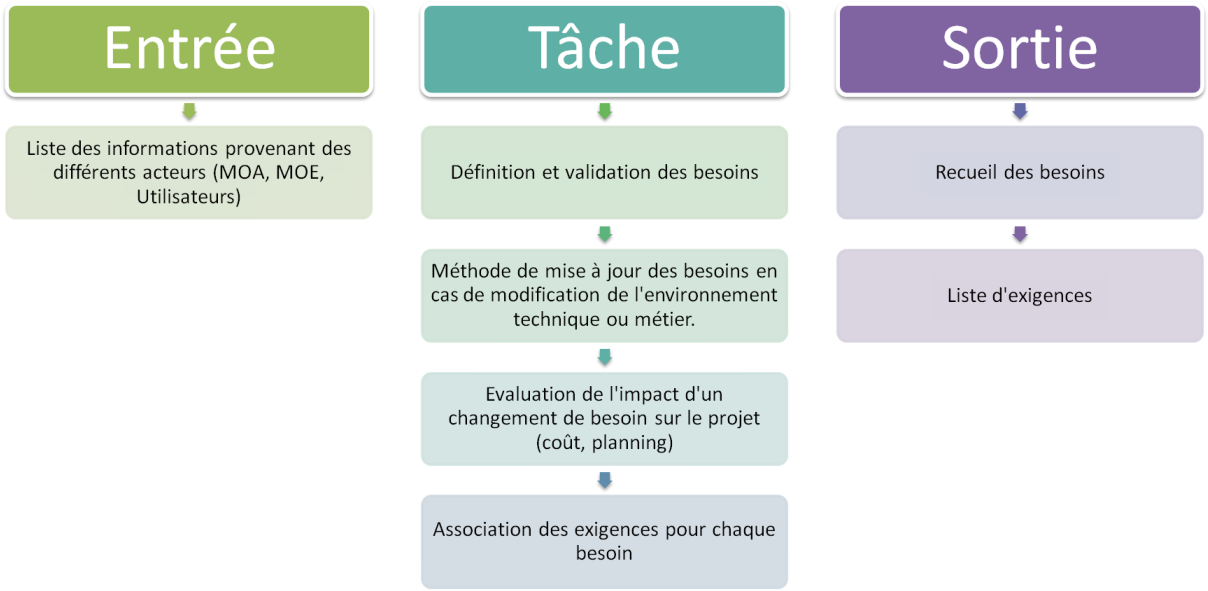


1. Recueil des besoins

Le chef de projet réunit l'ensemble des acteurs pour dresser la liste des besoins. Sont associés les initiateurs du projet pour traiter des éléments fonctionnels, les informaticiens pour les caractéristiques techniques et les utilisateurs pour l'ergonomie.

Au final les points suivants doivent êtres qualifiés :

- Les besoins du client,
- Les contraintes,
- Un état des lieux de l'existant,
- Une description des fonctionnalités principales de l'application à développer.



Une fois que le besoin est clairement exprimé, c'est le moment pour le chef de projet de se demander comment le produit va être développé. Cette réflexion va suivre les processus suivants :

- Conception générale de l'architecture système
- Analyse des exigences
- Conception détaillée du produit.

2. Conception générale

Ce document doit dresser une description de l'architecture fonctionnelle à travers des schémas et des explications claires.

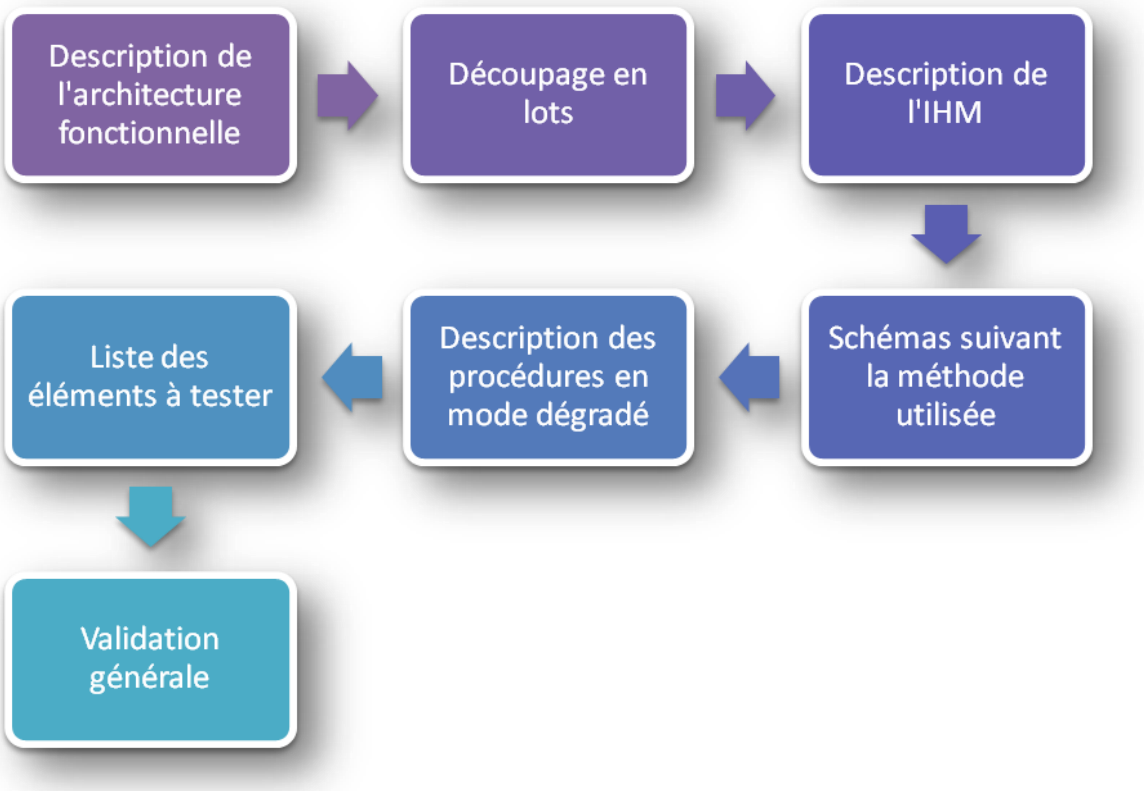
Si le produit à développer est d'une certaine complexité et s'appuie sur un SGBD, la méthode Merise sera privilégiée.

On y trouvera alors les différents modèles propres à cette méthode :

- Le Modèle Conceptuel de Données qui représente la structure du système d'information du point de vue des données.
- Le Modèle Conceptuel des Traitements où sont visualisés les éléments d'entrés et de sorties avec les événements qui les lient.

Si le développement utilise un langage orienté objet tel que Java, .Net ou C++, ce sera alors la méthode UML qui sera préconisée.

Quelque soit la méthode choisie, la rédaction du document de spécifications générales, pour être aboutie, devra suivre le process suivant :



Plus précisément, le document de conception générale doit aborder les sujets suivants :

- **Les règles de gestion :**

  Elles sont héritées des règles métier propre à l'entreprise. Ne sont listées que celles qui sont utilisées par l'application à développer. Une règle peut faire appel à un ou plusieurs processus fonctionnels.

- **Les processus fonctionnels :**

  On appelle processus fonctionnel une ou plusieurs tâches utilisées pour transformer une information d'entrée en une information de sortie. Ce processus peut être déclenché soit manuellement par l'utilisateur, soit automatiquement par un autre processus ou par un batch programmé par exemple.

  Exemple :

  - Information d'entrée : Valeur brute d'un produit en euros.
  - Processus fonctionnel : Ajout de la TVA à la valeur initiale.
  - Information en sortie : Valeur nette du produit.

- **Les méthodes de conversion de données :**

  Il arrive souvent que les données issues de certains systèmes doivent être retraités avant de pouvoir être utilisés par l'applicatif. Ces méthodes doivent être indiquées ici pour maitriser l'intégrité de l'information en entrée du produit à développer.

- **Les interfaces :**

  Les interfaces sont, ici, les moyens utilisés pour la communication entre le produit à développer et les systèmes existants au sein de l'entreprise et avec lequel l'applicatif a une interaction.

- **Les maquettes d'écran :**

  Les maquettes d'écrans permettent de se faire idée de ce peut donner le produit final. Elles doivent tenir de la charte graphique utilisée au sein de l'entreprise et montrer une certaine homogénéité. Ces maquettes peuvent évoluer en tenant compte des retours des utilisateurs, notamment pour améliorer l'ergonomie.

- **Fonctionnement en mode dégradé :**

  Si l'applicatif est appelé à être utilisé de manière continue, il faut prévoir des procédures si tout ou partie du logiciel tombe en panne et fournir une documentation claire pour les utilisateurs qui seraient confrontés à cet incident. Ce fonctionnement particulier est à penser en simultané avec la conception de l'applicatif lui-même car c'est un principe trop souvent oublié et parfois irrécupérable une fois la phase de développement lancée.

- **Tests d'intégration :**

  Il est important de prévoir le plan de test d'intégration au plus tôt. Ce plan doit contenir la validation de chacun des composants logiciels qui formeront l'application elle-même. Les tests d'intégration seront exécutés une fois les tests unitaires validés.

  Les tests d'intégration ont pour objectif de valider la qualité générale du progiciel. Ainsi, il n'est pas possible de prendre en considération tous les détails. L'architecture à tester doit donc être aussi simple que possible mais également modulaire afin de supporter les évolutions du projet qui ne manqueront pas de suivre.

  La conception générale doit permettre de mettre en place une structure de tests qui réponde aux conditions suivantes :

  - Les informations d'entrées et de sorties de l'application à valider doivent être clairement identifiées.
  - Les erreurs doivent faire l'objet d'une gestion particulière et doivent pouvoir être analysées dans un journal de logs dédié.
  - Le système à tester doit être découpé en modules fonctionnels clairs avec une définition précise des dépendances qui les lient.
  - L'intégration des différents composants du système doit respecter un ordre défini au plus tôt et refléter les tests associés.

- **Dossier d'exploitation :**

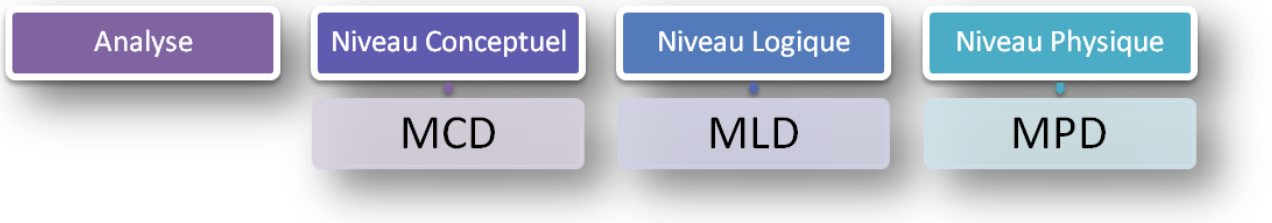
  Dès la phase de conception, il faut identifier les contraintes tant techniques que fonctionnelles auxquelles l'application sera confrontée. L'ensemble de ces contraintes sera centralisé dans le dossier d'exploitation.

3. Conception détaillée

Si la conception générale s'intéresse aux aspects fonctionnels du produit, la conception détaillée a pour objectif d'étudier l'architecture technique et répondre en particulier aux problématiques suivantes :

- Où sont stockées les données ?
- Quelle est leur volumétrie ?
- Quelles sont les informations d'entrées et de sorties ?
- Quels sont les traitements appliqués à ces informations ?
- Quels sont les critères de disponibilité de l'application ?
- Le produit doit-il respecter une norme de sécurité spécifique ?
- Quel est le langage utilisé pour le développement ?
- Quel est l'IHM utilisé
- Le produit demande-t-il un matériel particulier pour son utilisation régulière ?

L'analyse de l'architecture technique suit généralement la réflexion suivante :

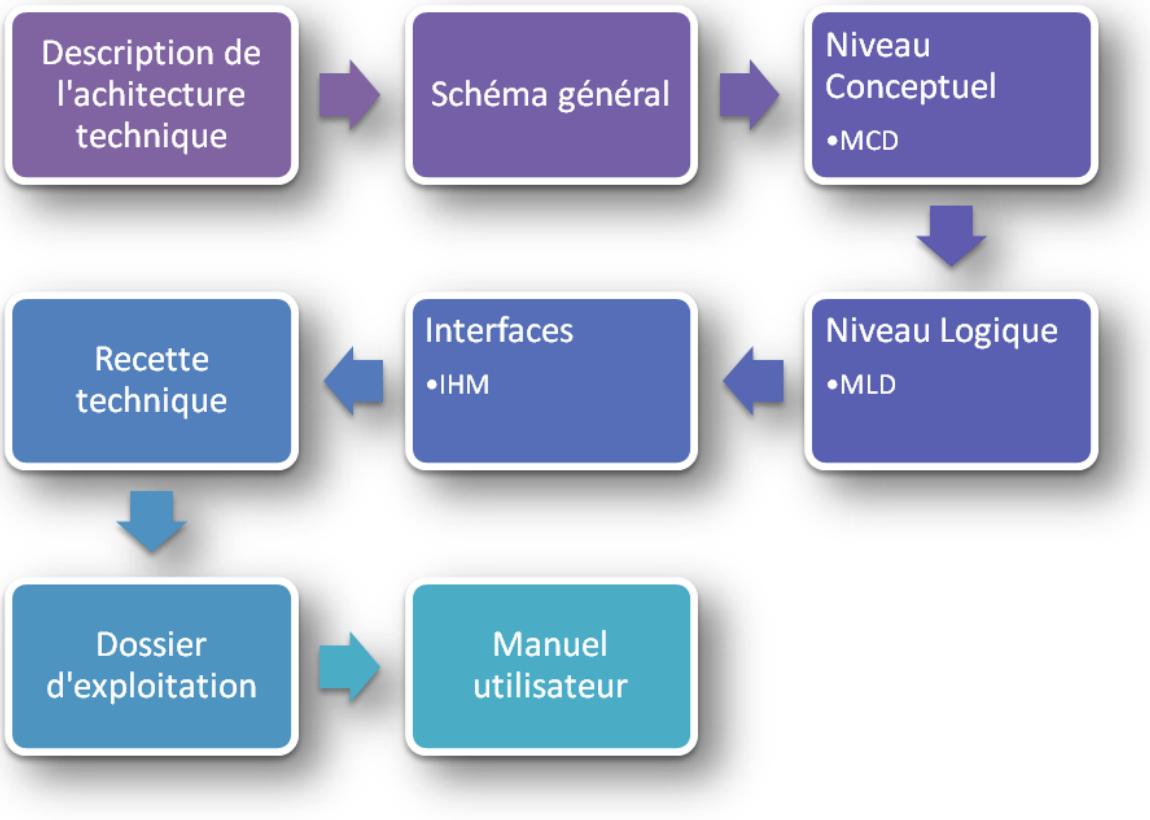


Ainsi les entités décrites dans le MCD sont traduites en tables dans le MLD où les liaisons inter entités deviennent des clés primaires et étrangères au sein de stables concernées.

Ainsi l'architecture technique doit couvrir les besoins suivants :

- Identification des chaînes de traitement,
- Liaison entre les interfaces,
- Identification des protocoles de communication,
- Quels sont les outils utilisés pour le développement du produit.

Le processus de création du document de conception détaillé va respecter l'analyse conduite précédemment :



Au final, le document de conception détaillé devra aborder les points suivants :

- **L'architecture technique :**

  Là seront décrits les composants logiciels et les traitements qu'ils renferment. A chacun des processus fonctionnels présentés dans les spécifications générales devra correspondre le moyen technique qui permet d'obtenir le résultat attendu par ce processus.

- **La technologie utilisée :**

  Cette partie est consacrée aux méthodes utilisées pour accéder aux données et aux interfaces utilisées :

  - Client lourd (application installée sur le poste utilisateur) ou client léger (application utilisant le navigateur Web).
  - Traitement des données en temps différé (les données affichées résultent d'un batch programmé) ou en temps réel (les calculs sont lancés à la demande du client)
  - Application client/serveur (les informations sont stockées sur une base de données distante) ou locale (les données sont stockées sur le poste utilisateur).

- **Les unités de traitement :**

  Un diagramme montrant les unités de traitement, c'est-à-dire les informations d'entrées et de sortie et les traitements qu'elles subissent devra être clairement représenté.

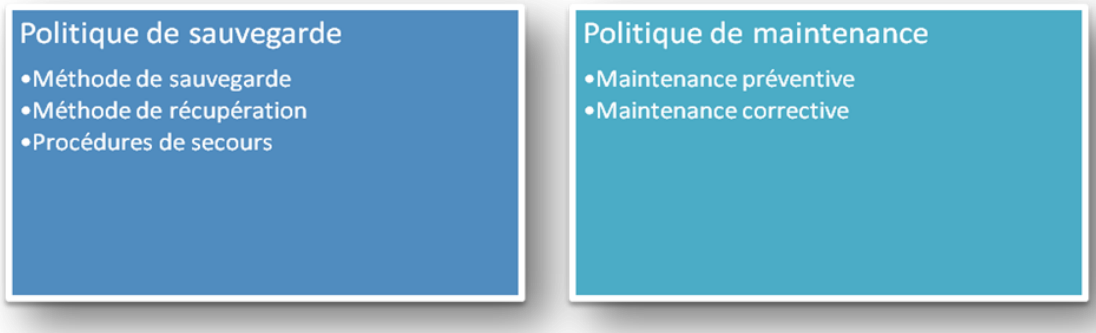
- **Les tests unitaires :**

  Chaque unité de traitement devra faire l'objet d'un test technique. Le succès de l'ensemble de ces tests permettra de valider l'application au niveau de sa conception.

- **Le dossier d'exploitation :**

  L'objectif de ce dossier est de fournir au personnel d'exploitation toutes les informations nécessaires à la mise en œuvre de l'applicatif. Ce dossier peut évoluer tout au long de la vie du logiciel afin de tenir compte des évolutions qui lui sont apportées.

  Un dossier d'exploitation peut être constitué des chapitres suivants :


Politique de sauvegarde
•Méthode de sauvegarde
•Méthode de récupération
•Procédures de secours
Politique de maintenance
•Maintenance préventive
•Maintenance corrective

V DEVELOPPEMENT ET VALIDATION TECHNIQUE

Une fois les spécifications validées, le chef de projet MOE peut organiser la phase de développement.

Elle comprend en fait trois étapes :

- Le codage du programme,
- Les tests unitaires,
- Les tests d'intégration.

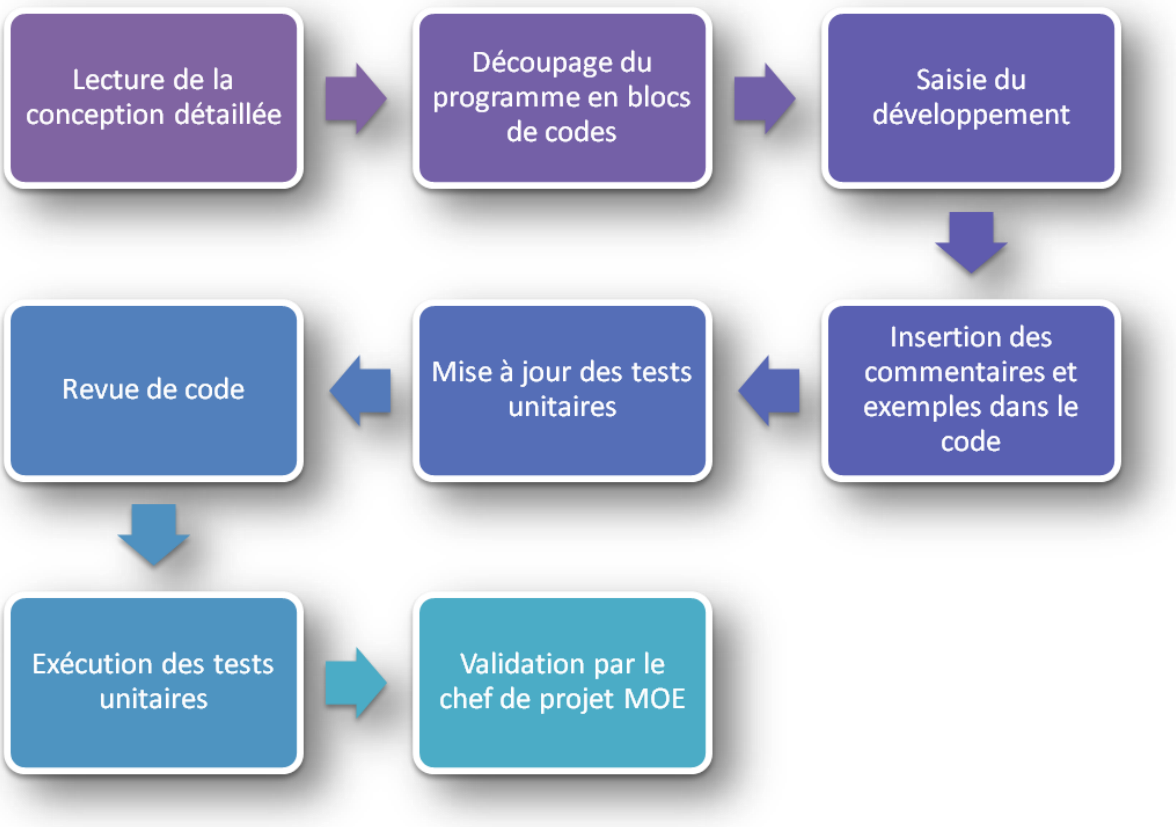
1. Développement de l'application

Il est une époque où l'on comptait uniquement sur le talent des développeurs pour fournir un produit acceptable. Au final, on obtenait parfois un logiciel de qualité inégale et faire évoluer le produit avec une équipe différente se révélait parfois très compliqué tant le code était emprunt de la méthode du développeur.

Aujourd'hui, fournir un logiciel correspondant à l'attente du client n'est plus suffisant. La qualité intervient désormais dans le code lui-même : il doit être clair, lisible, agrémenté de commentaires, d'exemples de résultats attendus et standardisé au mieux de manière à pouvoir à être repris par quiconque pour d'éventuelles corrections ou évolutions.

C'est le rôle du service Méthode & Qualité de dresser des normes que devront respecter les équipes de développement : qu'il s'agisse de règles de nommages pour les éléments de bases de données ou des syntaxes à préconiser en fonction de l'environnement client.

La phase de développement suit une succession d'étapes indispensables pour fournir un résultat qualitatif :



2. Tests unitaires

Les tests unitaires forment le premier moyen pour corriger liés au développement. Ils permettent de détecter les anomalies du type :

- Erreurs syntaxiques,
- Algorithmes de calcul incohérents,
- Cas non passants,
- Valeurs dépassant les limites établies,
- Appel à des objets distants inexistants,
- Code non optimisé.

3. Tests d'intégration

Les tests d'intégration ont pour but de vérifier si le produit ainsi développé répond aux exigences fonctionnelles. Un dossier de test d'intégration est alors dressé de manière conjointe entre l'équipe MOE et l'équipe de qualification.

Ce dossier puise dans les spécifications une liste exhaustive des éléments à tester tels que :

- La liste des fonctionnalités,
- Les règles de gestion,
- Les modules qui composent l'application.

A partir de ces informations, un plan de recette d'intégration est constitué. Ce plan doit comporter en outre :

- Des cas passants qui traversent l'intégralité de l'application,
- Des cas « à la marge » qui vérifient les débordements possibles,
- Des cas non passants qui éprouvent la stabilité de l'application et la cohérence des messages d'erreurs affichés,
- Les différents menus et liens associés,
- La bonne exécution des batchs.

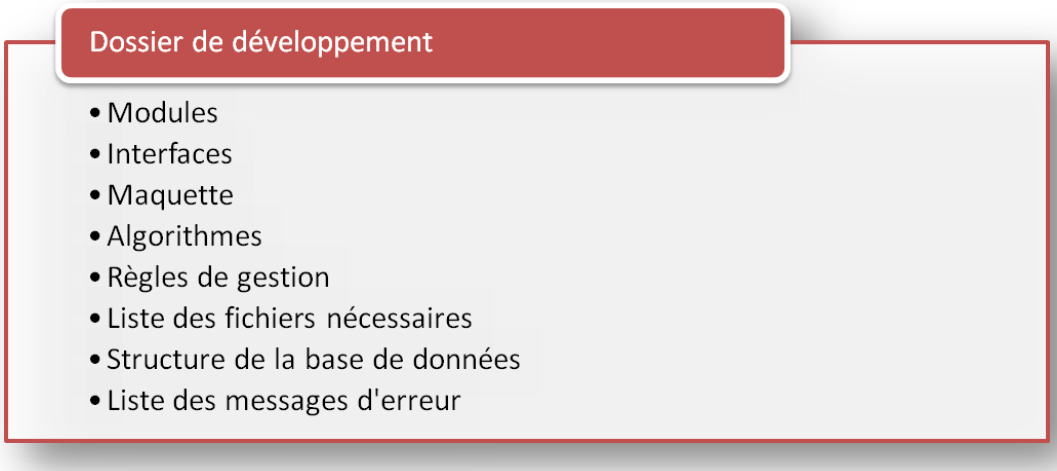
Pour conserver une mémoire des tests exécutés, l'équipe qualification rédige un cahier de recette qui sera tenu à jour jusqu'à la fin de la phase d'intégration.

4. Rédaction des livrables

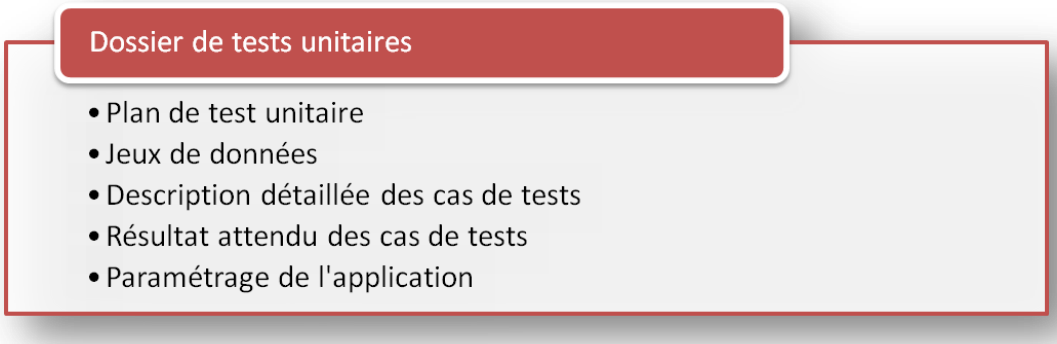
Les livrables sont des documents de référence où chacun pourra trouver les informations nécessaires à la bonne exécution de sa tâche. Il est donc important d'y consacrer le temps nécessaire pour fournir des dossiers de qualité.

A chaque étape de la phase d'intégration doivent être rédigés les livrables correspondants :

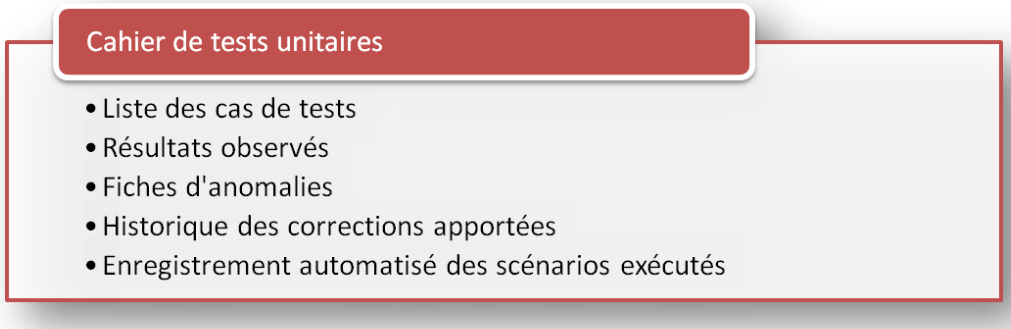
- **Développement :**
  Le dossier de développement doit permettre aux membres de l'équipe MOE de trouver toute l'information pour coder sereinement.



- **Tests unitaires :**
  Le dossier de tests unitaires est rédigé à l'issue de la validation des documents de conception dont il reprend les informations importantes.

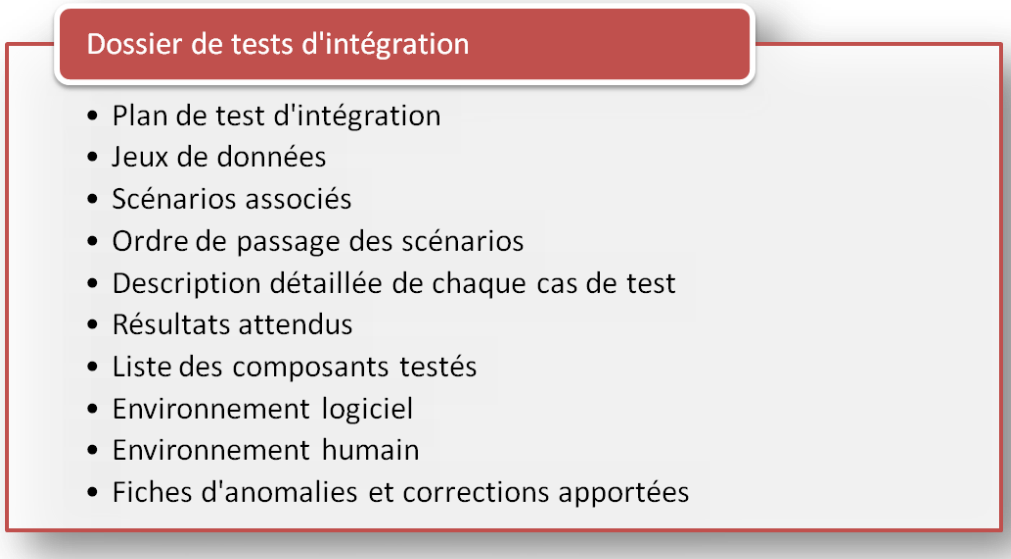


Le cahier de tests unitaires, lui, est conçu pour garder une trace des tests exécutés et des anomalies éventuelles. Il est donc tenu à jour tout au long de la phase des tests unitaires.

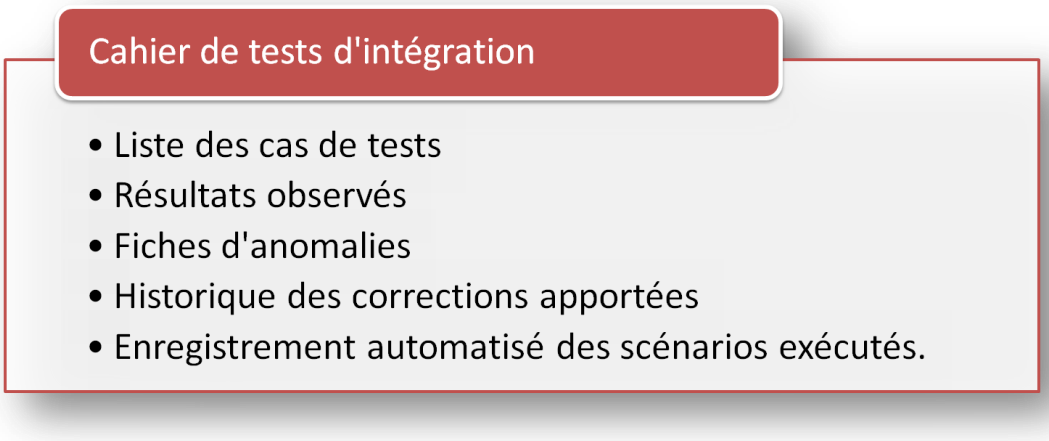


- **Tests d'intégration :**

A l'instar du dossier de tests unitaire, le dossier de test d'intégration est rédigé dès le début de la phase de développement en prenant ses informations dans les documents de conception.



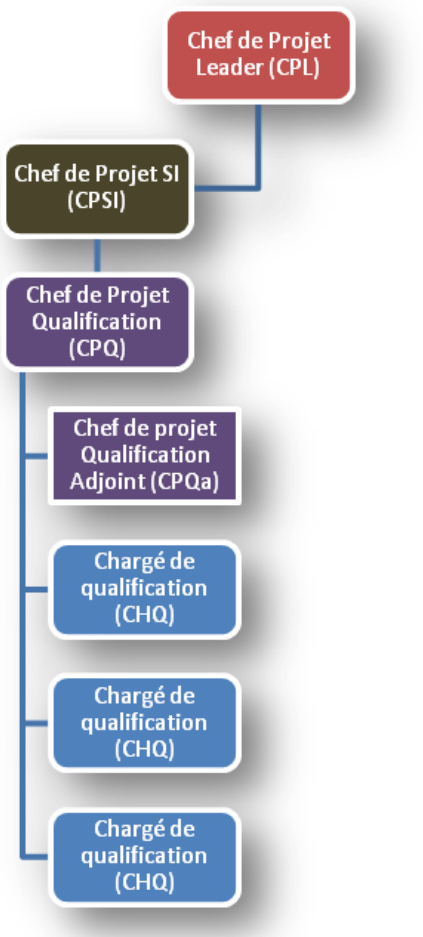
Le cahier de test d'intégration permet d'inscrire tous les événements liés à l'exécution des scénarios associés.



VI QUALIFICATION

La qualification prend une part importante dans le cycle de vie d'un logiciel. Son équipe est donc constituée d'une hiérarchie clairement identifiée où chacun a une tâche et une responsabilité précise.

L'organigramme de l'équipe au sein d'un grand compte peut être le suivant :



1. Répartition des tâches et responsabilités

- **Chef de Projet Leader (CPL) :**
  - Il identifie les parties prenantes et demande à chaque responsable une liste précise des exigences concernant l'application à développer.
  - Il organise le planning des différentes réunions d'avant-projet (ateliers, réunions techniques …) ainsi que les comités de pilotage qui surviennent tout au long du projet.
  - Il dirige les comités de pilotage et arbitre si des différents apparaissent entre certaines équipes.
  - Il supervise la rédaction de certaines documentations et les valide : expression de besoin, fiches de changement, compte-rendu de comité de pilotage.

- **Chef de Projet SI (CPSI) :**
  - Il s'assure que les solutions proposées sont cohérentes tant d'un point de vue technique que fonctionnel.
  - Il supervise les moyens informatiques mis à disposition du projet.
  - Il supervise et valide la rédaction des documents à sa charge : conception générale, conception détaillé.
  - Il s'assure que les exigences listées dans l'expression de besoin ont toutes été couvertes au sein du plan de recette.
  - Il est l'interlocuteur privilégié du CPQ pour toutes les questions techniques.

- **Chef de Projet Qualification (CPQ) :**
  - Il rédige les exigences de tests issues des documents de conception détaillée (technique et fonctionnelle).
  - Il pilote toute la qualification et travaille en relation directe avec les autres chefs de projets impliqués (MOE, MOA, Conduite du Changement).
  - Il gère le suivi des anomalies déclarées par l'équipe de recette, attribue les nouvelles anomalies aux personnes concernées et vérifie que les anomalies corrigées ont subi un test de validation.
  - Il assiste à toutes les réunions où la qualification prend une part de responsabilité.
  - Il gère le planning de chaque membre de l'équipe .
  - Il gère le budget alloué à la qualification et rend compte des dépenses au CPL.

- **Chef de Projet Qualification adjoint (CPQa) :**

Généralement, ce rôle est attribué à un chargé de qualification disposant d'une expérience conséquente. L'adoption des normes CMMI a alourdi les plannings des CPQ en tâches administratives et en réunions, aussi ces derniers aiment s'appuyer sur une personne de confiance pour les aider dans certaines responsabilités qui lui sont normalement dévolues. Le CPQa prend également à son compte la plupart des tâches dévolues aux chargés de qualification.

- Il vérifie que les exigences stockées dans l'outil de recette par le CPQ sont toutes liées par un cas de test et réciproquement
- Il prend en charge la rédaction de certains scénarios en fonction de leur complexité.
- Il aide les chargés de qualification dans la résolution des problèmes d'ordre professionnel.
- Il peut animer certaines réunions normalement dévolues au CPQ si celles-ci ne demandent aucun arbitrage.
- Il est chargé de vérifier que les anomalies déclarées par les membres de son équipe ont bien été prises en compte par les personnes que le CPQ a désigné et qu'elles sont en cours de correction.
- Selon ses connaissances, il peut également apporter des conseils techniques ou fonctionnels au CPQ.

- **Chargé de Qualification (ChQ) :**
  - Il est directement sous la responsabilité du CPQ.
  - Il est chargé de concevoir le plan de test
  - Il est chargé de concevoir les scénarios de test. Un scénario peut avoir pour objectif de tester une ou plusieurs fonctionnalités de l'application ou valider un critère technique particulier : performance, sécurité …
  - Il exécute les scénarios
  - Il est chargé de la rédaction et du suivi de ses anomalies.
  - A la réception d'une fiche de changement, il exécute le cas de test lié afin de vérifier la bonne correction de l'anomalie et de l'absence de non-régression.

Lorsque le Chef de Projet Qualification reçoit le feu vert pour commencer la phase de recette, sa première tâche est de vérifier si l'application livrée correspond effectivement à celle attendue : c'est la recevabilité. Une fois validée, la recette peut alors commencer.

2. Recevabilité

Bien que les tests d'intégrations soient censés fournir à l'équipe qualification une application techniquement propre, un CPQ avisé peut demander à la MOE une fiche de recevabilité où tous les éléments nécessaires au fonctionnement de l'application seront pointés.

Cette fiche réunit généralement :

- La liste des composants de l'application,
- La liste des interfaces,
- Quelques données de test et leur résultat attendus,
- La documentation utilisateur.

Tel un bordereau de livraison, le CPQ cochera chacune des lignes et ne signera que si l'ensemble des éléments listés a fait l'objet d'une vérification,

La phase de recette proprement dite pourra alors débuter.

3. Préparation de la recette

L'équipe de qualification se lance dans une phase de préparation des tests, En premier lieu, le périmètre des tests est dressé en accord avec les exigences.

Ce périmètre englobera, à minima, les vérifications suivantes :

- Les règles de gestion,
- Les algorithmes de calcul,
- Toutes les opérations susceptibles d'êtres passées par l'utilisateur,
- Les cas non passants et les messages d'erreur,
- L'ergonomie de chaque interface,
- La communication avec les autres systèmes.

Une fois ce périmètre correctement cerné, le plan de test est alors monté. Chacun des cas de test doit être lié à une exigence pour le moins. Une fois le plan intégré dans l'outil adéquat (exemple : HP Test Designer), l'équipe qualification doit construire les scénarios de test.

Il est parfois difficile de faire la distinction entre un plan de test et un scénario, Un plan est une liste de cas de tests rédigés de manière la plus générique possible en accord avec les exigences, Un scénario reprend un ou plusieurs cas de tests issu du plan pour vérifier une fonctionnalité précise.

4. Stratégie de test

La stratégie de test est le document structurant car elle organise tout le cycle de la qualification. A partir des besoins du client et des contraintes tant techniques que fonctionnelles, elle permet au final de découper les périodes de test et de lister les exigences, points d'entrée pour la rédaction du plan de test.

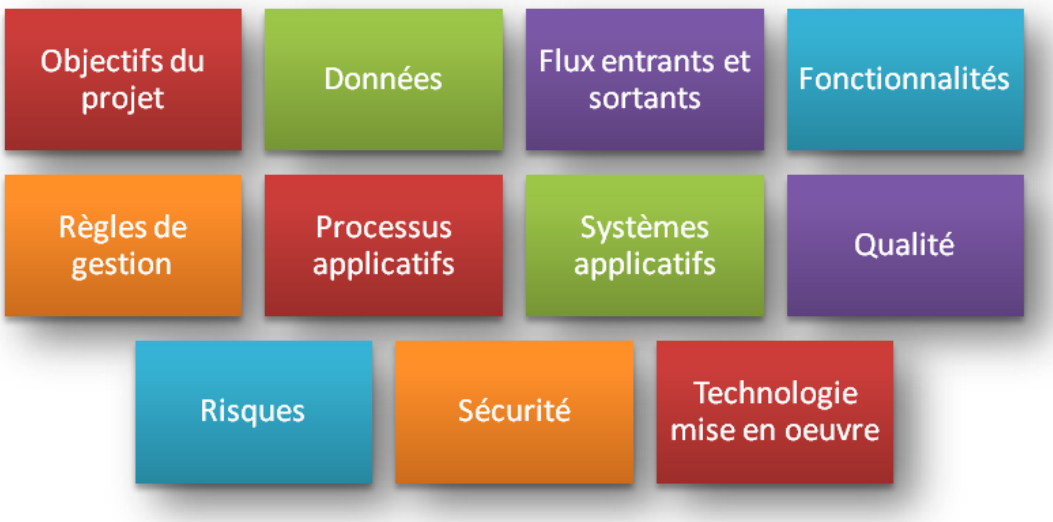
Toute la phase de qualification va donc être ordonnancée en fonction de la stratégie de test

Pour produire une stratégie de test, le Chef de Projet Qualification s'appuie sur 3 documents essentiels :

- **L'effort de test théorique :**

  Ce document définit le périmètre théorique de la phase de qualification. Il s'appuie sur tous les écrits rédigés en amont (expressions de besoin, conceptions techniques et fonctionnelles …) et sur les comptes-rendus de réunions avec les différents acteurs du projet. En conclusion, il dresse un planning de test théorique.

  Voici une liste des informations à recueillir :



Ce document définit le périmètre théorique de la phase de qualification. Il s'appuie sur tous les écrits rédigés en amont (expressions de besoin, conceptions techniques et fonctionnelles …) et sur les comptes-rendus de réunions avec les différents acteurs du projet. En conclusion, il dresse un planning de test théorique.

- **La stratégie de test réelle :**

  Ce document détaille les contraintes appliquées au projet : elles sont d'ordre technique, environnemental ou fonctionnel. Une fois validés par les différents intervenants, ce document aboutit à une révision du planning initial.

- **Le découpage de la qualification :**

  La phase de qualification est organisée en étapes distinctes : cycles, campagnes … Ce découpage donnera lieu à des bilans et sera visible de l'outil de gestion des tests, tel que Test Director.

  Le schéma ci-dessous explicite les étapes nécessaire à la rédaction de la stratégie de test jusqu'à la création du plan de test.

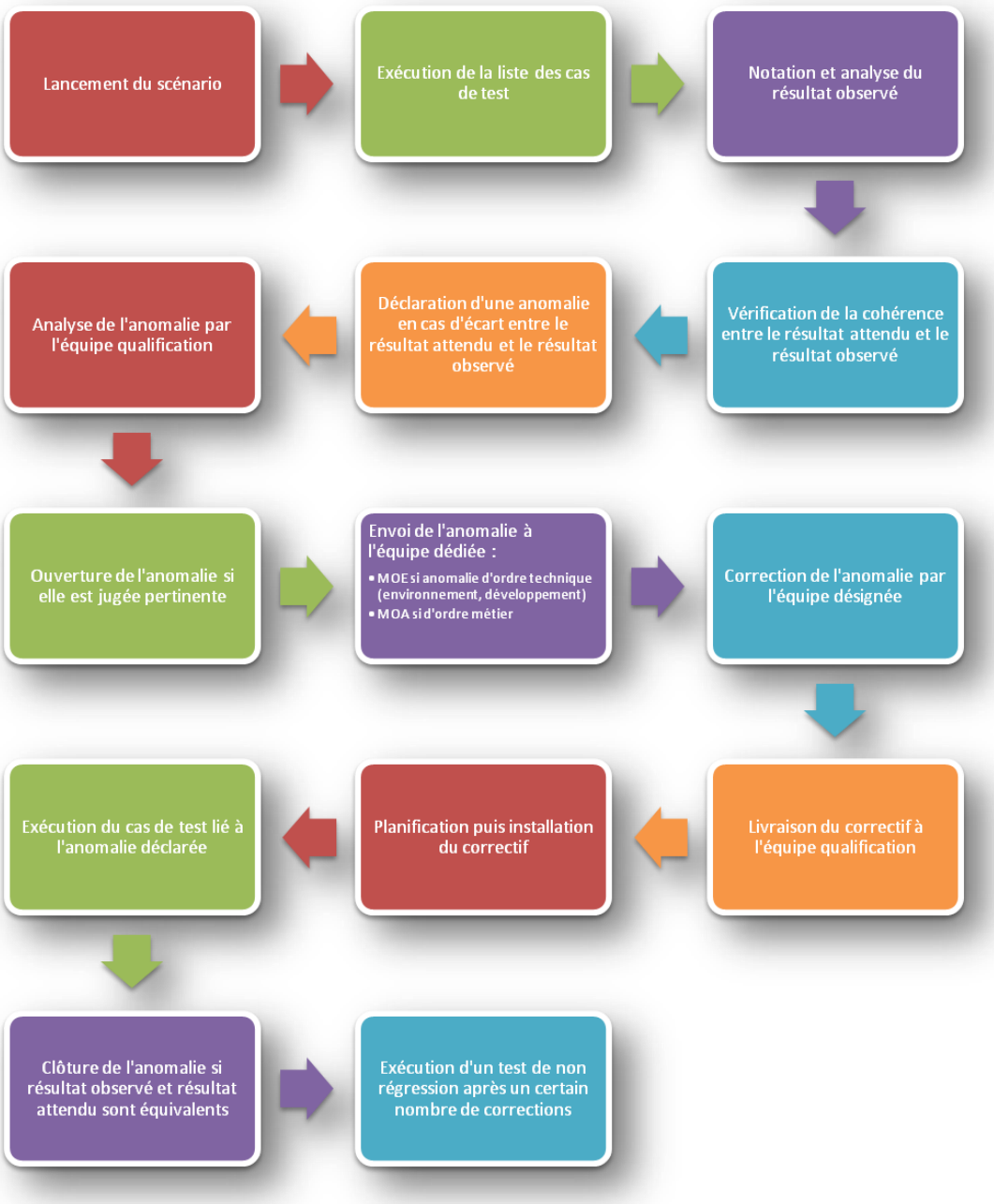

Etape 1 : Analyse documentaire
• Expressions de besoins
• Spécifications techniques
• Spécification fonctionnelles
• Maquette
• Autres documents ...
Etape 2 : Classification des exigences
• Exigences techniques
• Exigences métier
• Criticité des exigences
Etape 3 : Exigences de tests
• Exigences de tests techniques
• Exigences de tests fonctionnels
• Exigences de tests transverses
Etape 4 : Définition des moyens
• Moyens techniques
• Moyens humains
• Budget et planning
Etape 5 : Effort de test théorique
• Fonctionnalités
• Processus
• Exigences
• Priorités
• Charges

La rédaction de la stratégie doit intervenir le plus tôt possible dans le cycle de vie du logiciel, dans l'idéal dès la phase de conception générale. Sa validation doit intervenir avant le début de la phase de qualification afin que l'équipe concernée puisse intégrer un plan de test cohérent et commencer la rédaction des scénarios de test.

5. Exécution de la recette

Une fois les scénarios organisés, les jeux de données livrés par l'équipe MOE, le Chef de Projet Qualification peut alors, soit laisser son équipe exécuter les tests, soit dédier ce travail à des tierce personnes, généralement un panel sélectionné parmi les utilisateurs finaux.

L'exécution de la recette s'organise ainsi :



Le Chef de Projet Qualification rédige un procès verbal de fin de recette lorsque les conditions nécessaires sont réunies :

- Toutes les anomalies sont closes,
- Les scénarios ont été intégralement déroulés,
- Les tests de non régression ont montré que le produit est resté cohérent à l'issue des corrections,
- L'intégrité des liens entre le plan de test et les exigences a été respectée.

Ce PV est finalement signé par toutes les parties prenantes et permet alors au Chef de Projet de passer la suite soit directement soit à l'équipe de conduite du changement, soit directement au client.

VII ACTIVITE DE TEST

1. Présentation

Il ne suffit pas de bien maitriser l'environnement technique et métier du produit à recetter, encore faut-il poser la bonne question … C'est-à-dire, dans le métier de la qualification, utiliser la bonne méthode pour rédiger le cas de test avec la plus grande clarté possible afin d'éviter tout problème de compréhension de la part du testeur.

Le métier du test a longtemps été dénigré, au mieux exploité à minima dans les projets de développement en tant que tests unitaires. Heureusement l'intégration de normes et de modèles au sein des entreprises ont montré l'intérêt du test et lui laissent désormais une place entière dans la gestion de projet.

En pratique, le métier du test réunit les activités suivantes :

- **Planification** :

  Non seulement, il faut prévoir un personnel suffisant, qualifié et attribuer à chacun des tâches en adéquation avec les compétences mais il faut également gérer les priorités des tests et, surtout maitriser le surcout amené par la gestion des anomalies.

- **Spécifications** :

  Il s'agit de mettre en place un classement des tests en fonction de leur type (fonctionnel, technique) et d'exprimer les besoins en terme d'environnement techniques et humains.

- **Conception :**

  La tâche de conception du plan de test et des scénarios est primordiale. Elle demande un effort d'analyse et de synthèse important aussi il est nécessaire d'accorder un délai suffisant et s'assurer de la disponibilité de tous les éléments : documents, interlocuteurs …

  Il faut également prévoir les jeux de données nécessaires à la bonne exécution des scénarios.

- **Contrôle de l'attendu :**

  L'exécution d'un test demande une rigueur dans l'analyse lorsqu'un écart est constaté entre le résultat attendu et le résultat observé.

- **Traçabilité** :

  Une matrice de traçabilité doit être construite et doit permettre de vérifier que chaque exigence a fait l'objet d'au moins un test (*complétude*) et que chaque test est effectivement lié à une ou plusieurs exigences (*efficacité*).

Pour mener à bien ces différentes tâches, un spécialiste du test doit posséder des qualités particulières. On peut citer, par exemple :

- De la **curiosité** et de l'**imagination** : en effet, ces deux compétences sont nécessaires pour projeter des cas d'erreur d'un progiciel alors qu'il n'est pas encore développé !
- Une sorte de **pessimisme** car le testeur part du constat que tout développement, aussi bien mené soit-il, ne peut être exempt de défauts.
- Avoir un **œil neuf** sur l'environnement client peut permettre de déceler des failles ou imaginer des situations qu'une personne depuis longtemps immergé dans le métier n'aurait pu voir.
- Etre très **attentif aux détails** qu'ils soient d'ordre technique ou fonctionnel car, souvent des défauts voire des oublis peuvent s'y cacher.
- Faire preuve d'un **bon relationnel** est vital car le testeur a besoin de construire un réseau autour du projet afin de pouvoir récupérer les informations qui peuvent lui manquer principalement pour la phase d'analyse des besoins que pour la phase de conception du plan de test.
- Une facilité à **s'adapter à ses interlocuteurs** : le métier du test est transverse et amène le professionnel à débattre tant avec les experts techniques qu'avec le client, plus orienté métier.

2. Les niveaux de test

On le sait maintenant, la création d'un logiciel comprend 3 grandes étapes :

- La phase de conception des différents modules qui vont composer le logiciel,
- La phase d'intégration qui va permettre l'interaction des différents modules développés,
- La phase de déploiement du produit fini dans un état propre à son exécution.

A chacune de ces étapes correspond un niveau de test particulier :

3. Liste de tests

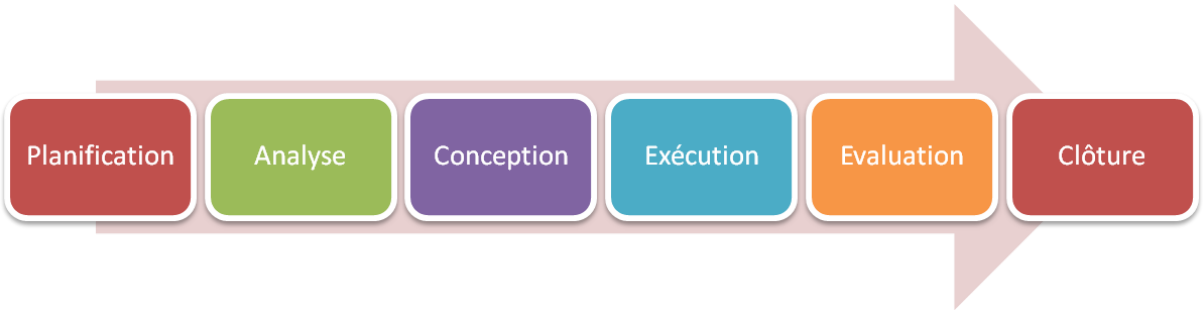
Si le test unitaire est une phase importante dans le développement, l'intégration de l'équipe qualification dès l'amont du projet permet d'éviter de nombreux écueils grâce à des interventions stratégiques en corrélation avec l'évolution du chantier.

En effet, l'activité de test suit la chronologie du cycle de vie du logiciel. Le tableau ci-dessous dresse une liste des tests qui accompagne le projet :

Phase	Type de test
Analyse des exigences	- Revue du modèle général
Analyse des documents de conception :	- Test du prototype - Test des fonctionnalités - Test des composants
Analyse du développement :	- Test de la boite noire - Test aux valeurs limites - Revue du code - Couverture des tests - Test des méthodes - Tests des chemins et liens internes et externes - Test de la boite blanche
Analyse de l'environnement système :	- Test du fonctionnement global - Test d'installation sur un environnement vierge - Test de montée en charge - Test de cohabitation avec les applications voisines

L'activité de test suit un cheminement précis où chaque étape est cruciale : de l'organisation de l'équipe qualification à la conception des tests, de l'exécution des tests à leur clôture, une planification maitrisée est nécessaire pour respecter les engagements.

Ainsi le cycle du test doit suivre le cheminement suivant :



4. Cycle du test

De l'organisation des tests en amont du projet à leur clôture, de nombreuses tâches sont nécessaires pour mener à bien l'activité.

- **Planification :**
  - Identification des ressources techniques et matérielles
  - Identification des objectifs de test : portée, risques, planning alloué …
  - Identification des types de test (techniques, fonctionnels, automatiques …) en corrélation avec les acteurs du projet (MOE, MOA …)
  - Planification des tâches :
    - Analyse
    - Conception
    - Exécution
    - Correction des anomalies (détection, rejeu des tests associés) : il faut ajouter de 10 à 20% à l'estimation générale pour gérer cette partie.
    - Identification des critères de réussite

- **Analyse :**
  - Analyse des informations mises à disposition : spécifications, architecture technique, interfaces …
  - Rédaction et classification des exigences de test

- **Conception** :
  - Mise en place de l'environnement de test : outils, personnel, accès aux environnements ...
  - Conception du plan de test
  - Conception des scénarios de test
  - Vérification de la couverture des exigences par les cas de test adéquats.
  - Création d'un cahier de test

- **Exécution :**
  - Organisation des équipes de tests : formation à l'outil de test et aux objectifs attendus s'il s'agit d'utilisateurs finaux.

- Organisation de la campagne de test : répartition des tâches d'exécution manuelle, planification des horaires de lancements des batchs automatiques.
- Exécution des scénarios de test.
- Analyse des tests exécutés : comparaison entre résultats attendus et résultats observés.
- Gestion des anomalies : déclaration, suivi, exécution des cas de tests liés après correction, clôture.
- Mise à jour du cahier de test

- **Evaluation** :
  - Synthèse des résultats des tests :
    - Pourcentage des tests passés avec succès.
    - Pourcentage des tests en erreurs.
    - Pourcentage des tests non passés.
    - Classification des anomalies encore ouvertes par criticité.
  - Vérification de l'adéquation entre la synthèse et les critères de sortie préconisés.
  - Rédaction d'un rapport de synthèse à destination des acteurs du projet. Il sera joint au cahier de tests.

- **Clôture** :
  - Rédaction de fiches de demandes d'évolution pour les anomalies restantes.
  - Revue des tests et mise à jour du cahier de tests
  - Archivage de l'environnement de tests
  - Proposition d'amélioration pour l'activité de tests en vue des problèmes rencontrés
  - Signature de la phase d'acceptation du logiciel entre les parties prenantes.

5. Familles de test

Les tests ont des objectifs précis : vérifier un aspect fonctionnel ou technique de l'application à un niveau donné du développement. Il est intéressant de les regrouper dans des familles distinctes afin de mieux cerner leur utilité.

- **Tests des composants logiciels :**
  - Tests de fonctionnalités.
  - Tests d'éléments techniques de l'application.

  Ces tests sont issus des spécifications fonctionnelles et techniques avec le support des équipes de développement. Les anomalies découvertes ici sont corrigés une par une dans les plus brefs délais.

- **Tests d'intégration :**
  - Tests d'interaction entre les composants.
  - Tests d'enchaînement de transactions.
  - Tests de non régression

  Une fois que l'équipe de développement a validé l'ensemble des modules de l'applicatif, Celui-ci est packagé pour une livraison.

  Commence alors les tests d'intégration qui vont permettre de valider que les différents modules développés fonctionnement correctement ensemble.

- **Tests de conformité ou tests système :**
  - Tests de comportement de l'application basé sur des cas d'utilisation généraux
  - Tests type « boite noire » pour vérifier les spécifications fonctionnelles
  - Tests type « boite blanche » sur les menus ou autres méthodes de navigation au sein de l'application.

  Les tests système sont mis en œuvre lorsque l'application est en fin de développement. La finalité de ces tests est de montrer le comportement de l'application dans un usage proche de la production.

- **Tests d'acceptation :**
  - Tests avant déploiement sur des caractéristiques techniques
  - Tests sur des opérations de maintenance

- Tests de sauvegarde et de restauration
- Tests sur des failles de sécurités identifiées
- Tests d'utilisation

Les tests d'acceptation n'ont pas pour objectif principal de lever des anomalies. Alors que le produit est en passe d'être livré, on ne devrait d'ailleurs plus découvrir de nouvelles défaillances. Le but est plutôt d'analyser le comportement de l'application ou sein de l'environnement client ou dans un environnement proche de la production.

Ces tests peuvent se présenter sous la forme de :

- Tests utilisateurs où un panel de personnes désignées par le client donnera son avis sur l'outil à travers des tests transversaux.
- Tests opérationnels où l'on effectue des manipulations spécifiques : sauvegarde, restauration, crash tests …
- Tests Alpha et Beta où une large gamme d'utilisateurs est mise à contribution pour manipuler librement le produit.

- **Tests fonctionnels :**
  - Tests du comportement extérieur du produit
  - Tests de sécurité : stabilité de l'application face à des éléments logiciels ou matériels de sécurité (pare-feu, antivirus, proxy …)

Ces tests permettent de mettre à l'épreuve les fonctionnalités décrites dans les spécifications ou tout autre logiciel ou matériel dédié à la sécurité.

- **Tests non fonctionnels :**
  - Tests de performances.
  - Tests de comportement et de stabilité : fuites de mémoire, surcharge, connexions simultanées etc.
  - Tests d'interopérabilité avec les applications clientes.

Ces tests se situent à différents niveau du cycle de vie du logiciel. Ils permettent d'éprouver les éléments techniques et de vérifier leur adéquation avec les contraintes du client.

- **Tests structurels :**
  - Tests de couverture
  - Tests de type Boite blanche

Ces tests ont pour but d'évaluer précisément le niveau de couverture des exigences. Elles peuvent mesurer l'avancement des tests ou du développement qu'il s'agisse de l'application dans son ensemble ou d'une fraction particulière du produit.

- **Tests post modification :**
  - Tests de conformité
  - Tests de régression

Après détection et correction d'une anomalie, qu'elle soit d'ordre fonctionnelle ou technique, il faut valider les modifications par un test, généralement celui qui a permis sa levée. Ce type de test est du ressort de l'équipe de développement.

Les tests de régression sont inhérents à toute modification. Ils permettent notamment de :

- Tester les anomalies corrigées et vérifier que la correction a bien résolu le problème sans créer de problèmes sous-jacents.
- Supprimer les tests redondants.

- **Tests de maintenance :**
  - Tests après mise à jour
  - Tests après migration
  - Tests de l'environnement après suppression de l'application.

Une fois en production, le produit est amené à subir des évolutions, des changements de matériel et, au final, une mise à la retraite pour être remplacé par une nouvelle version ou un applicatif concurrentiel. A chacune de ces modifications, les tests de maintenance permettent de vérifier que l'applicatif lui-même est toujours conforme aux attentes à travers des scénarios opérationnels.

- **Tests statiques :**
  - Revue du code
  - Revue des exigences
  - Revue des spécifications
  - Revue des scénarios de test

- Analyse statique

Derrière le terme statique se cache l'idée d'analyser le produit sans l'exécuter. En pratique, on rechercher à vérifier la cohérence des informations périphériques à l'application.

Lorsque cette opération est manuelle, on parle de revue ; automatique, il s'agit alors d'**analyse statique**.

L'intérêt principal de ce type de test est de détecter le plus tôt possible des anomalies liées aux livrables et d'y apporter les corrections nécessaires avant leur prise en main par les équipe de développement.

Pour mener à bien une phase de revue, il faut :

- Sélectionner éléments à analyser
- Parcourir en profondeur les éléments désignés.
- Rédiger un document de synthèse où seront notées les anomalies potentielles.
- Organiser une réunion avec les principaux acteurs des projets et commenter les défaillances probables.
- Faire des recommandations pour la correction des défauts. Ecouter les propositions.
- Demander aux auteurs des informations analysées d'effectuer les corrections.
- Vérifier les nouvelles versions des informations corrigées.

Un document de revue peut être rédigé est contenir plusieurs chapitres, chacun d'eux étant dédié à une analyse statique précise.

Par exemple, il peut contenir les chapitres suivants :

- Revue de code :

  Le code est divisé en blocs fonctionnel et il est commenté par l'équipe technique.
- Revue technique :
  - Réunion de préparation à la revue technique
  - Liste des anomalies et historique de leur correction
  - Rapport de la revue
  - Propositions d'évolutions, commentaires divers
- Revue fonctionnelle :
  - Revue menée par un tiers désigné par le client.
  - Inspection basée sur des check lists avec des critères d'entrée et de sortie
  - Participation des différents acteurs du projet
  - Propositions d'améliorations
- Revue de projet :
  - Synthèses des réunions importantes

- Rapport de revue

En ce qui concerne l'analyse de code, par exemple, il existe des outils qui permettent de rechercher des défauts de manière automatisée. Ces outils se basent sur des règles définies par les développeurs et traquent les anomalies de type :

- Variables avec des valeurs incohérentes ou indéfinies
- Variables non utilisées
- Code inutilisé
- Respect des normes de codage
- Problèmes de syntaxe
- Failles de sécurité

6. Méthode de test général

Les techniques proposées ici peuvent être utilisées tant pour vérifier un point fonctionnel du produit que pour valider un critère technique.

- **Test par affirmation :**

  L'objectif de ce type de test est de vérifier la validité d'un aspect du produit en accord avec l'exigence associée.

Exemple :

Type de test :	- Fonctionnel
Description :	- Le nom de la personne connectée sur l'application doit voir son nom s'afficher dans un espace dédié
Paramètres en entrée :	- Login et mot de passe d'un utilisateur ayant des droits d'accès à l'application
Résultat attendu :	- Le nom et le prénom de l'utilisateur apparait dans un cadre en haut à droite de la page d'accueil

- **Test par négation :**

Ce test permet d'exploiter les failles de l'application, c'est-à-dire il doit vérifier le son comportement en effectuant des manipulations hors du fonctionnement prévu.

L'emploi de ce type de test est particulièrement important les spécifications manquent de précision sur un point particulier.

Exemple :

Type de test :	Technique
Description :	Vérifier le comportement de l'annuaire lorsque l'on lance une recherche sans donner de critères.

Paramètres en entrée :	- Validation sans critères
Résultat attendu :	- un message explicite apparaît obligeant l'utilisateur à saisir au moins 1 critère.

7. Tests techniques

Les techniques présentées ici permettent de vérifier le bon fonctionnement des composants internes du logiciel, du point de vue des équipes de développements et responsables système.

- **Test de la boite blanche :**

  Cette technique a pour but d'analyser la structure interne d'un composant ou d'un module particulier de l'application à travers des tests précis.

Exemple :

Grâce à un test de couverture, on souhaite vérifier que l'instruction est parcourue au moins une fois avec le paramètre d'entrée adéquat.

Type de test :	Technique
Description :	chaque condition présente dans le code est exécutée selon le jeu de donnée en entrée.
Paramètres en entrée :	Cas 1 : x = 1 Cas 2 : x = 5 Cas 3 : x = -2
Instruction à tester :	If x> 0 and x < 5 Then print « la valeur recherchée est dans le bon intervalle » Else print « La valeur recherchée ne correspond pas aux critères »
Résultat attendu :	Cas 1 : « la valeur recherchée est dans le bon intervalle » Cas 2 : « La valeur recherchée ne correspond pas aux critères » Cas 3 : « La valeur recherchée ne correspond pas aux critères »

8. Tests fonctionnels

L'approche des tests proposés ici est orientée côté fonctionnel afin de vérifier la cohérence du produit avec les exigences métier.

- **Test de la boite noire :**

  Cette technique a pour but de vérifier le comportement du logiciel sans prendre en considération sa structure interne. L'objectif est donc de tester les différentes

fonctionnalités précisées dans les exigences en se basant sur le rapport entre les données fournies en entrée et les informations en sortie.

Exemple :

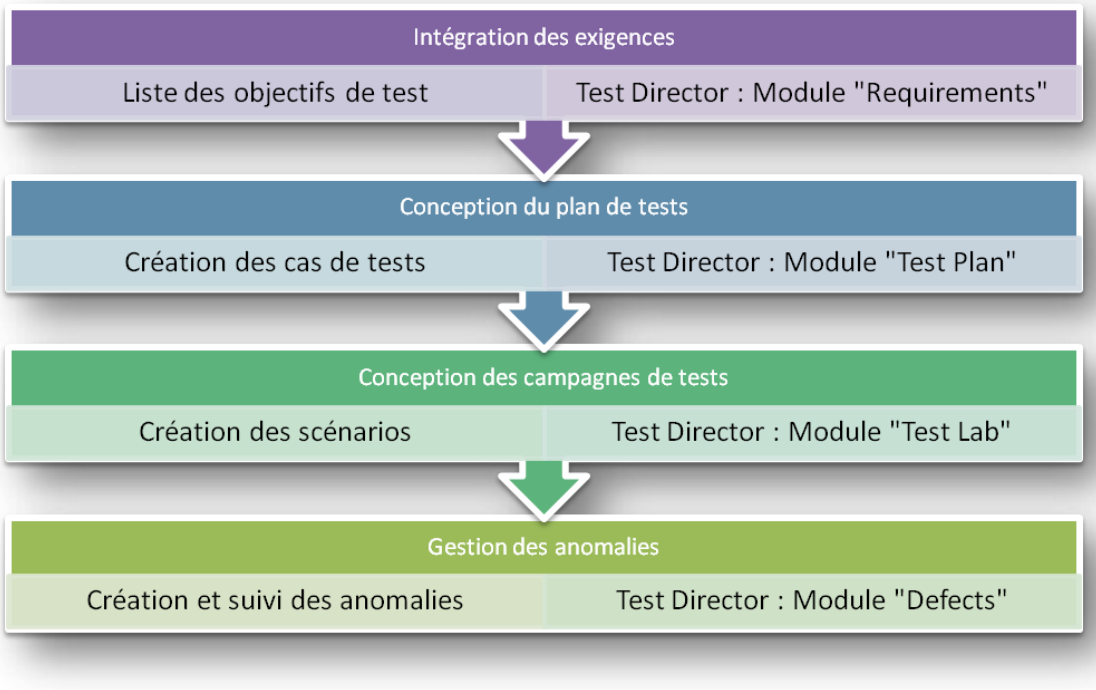
Type de test :	Fonctionnel
Description :	Vérifier les informations affichées en sortie en fonction d'un nom et d'un prénom fournis au moteur de recherche de l'annuaire.
Paramètres en entrée :	Nom : Berlitz Prénom : Jean-Baptiste
Résultat attendu :	Le moteur trouve la personne désirée et fournie les données suivantes : Nom : Berlitz Prénom : Jean-Baptise Service : Relations commerciales Poste : 2164 E-mail : jb.berlitz@fuseo.com

VIII TEST DIRECTOR : UN OUTIL DE GESTION DES TESTS

1. Présentation

HP Test Director est un outil qui permet de gérer de A à Z une recette applicative. En effet, de l'intégration des exigences à la gestion des anomalies, il offre une vue en temps réel de l'état du projet et permet à chaque membre de l'équipe de participer ou de simplement de visualiser un élément précis du projet ou d'en tirer des statistiques.

Le schéma ci-dessous illustre l'interaction entre Test Director et une recette logicielle :



Pour organiser la phase de qualification, le Chef de Projet se base sur la stratégie de test. Celle-ci est établie grâce à :

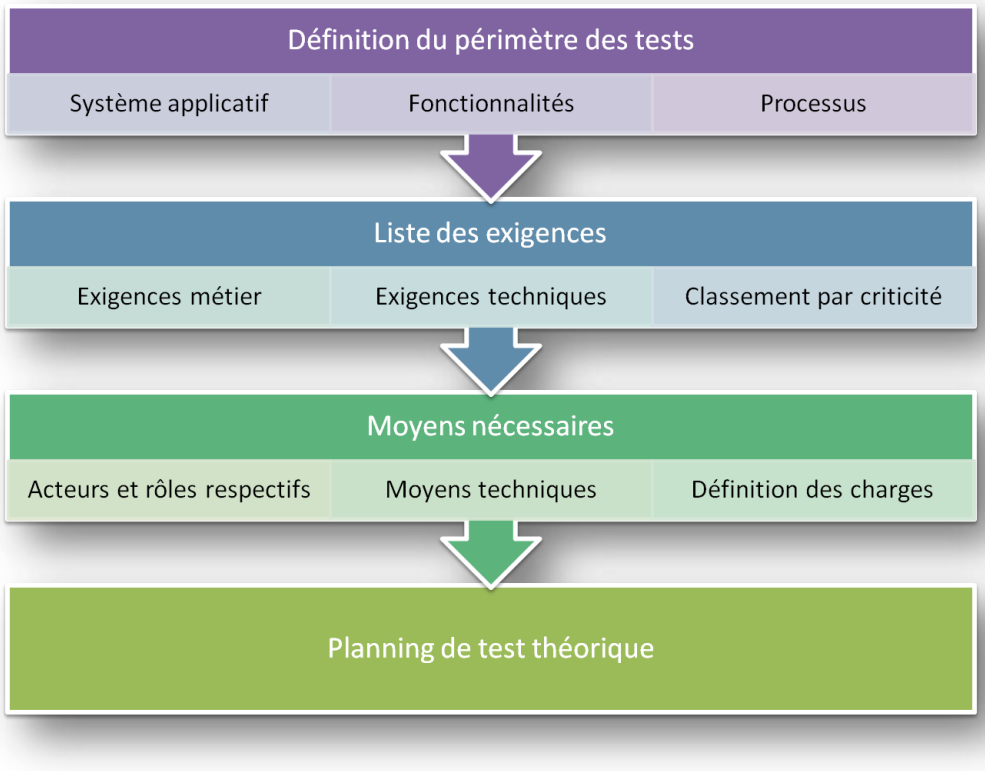
- **L'effort de test théorique :**

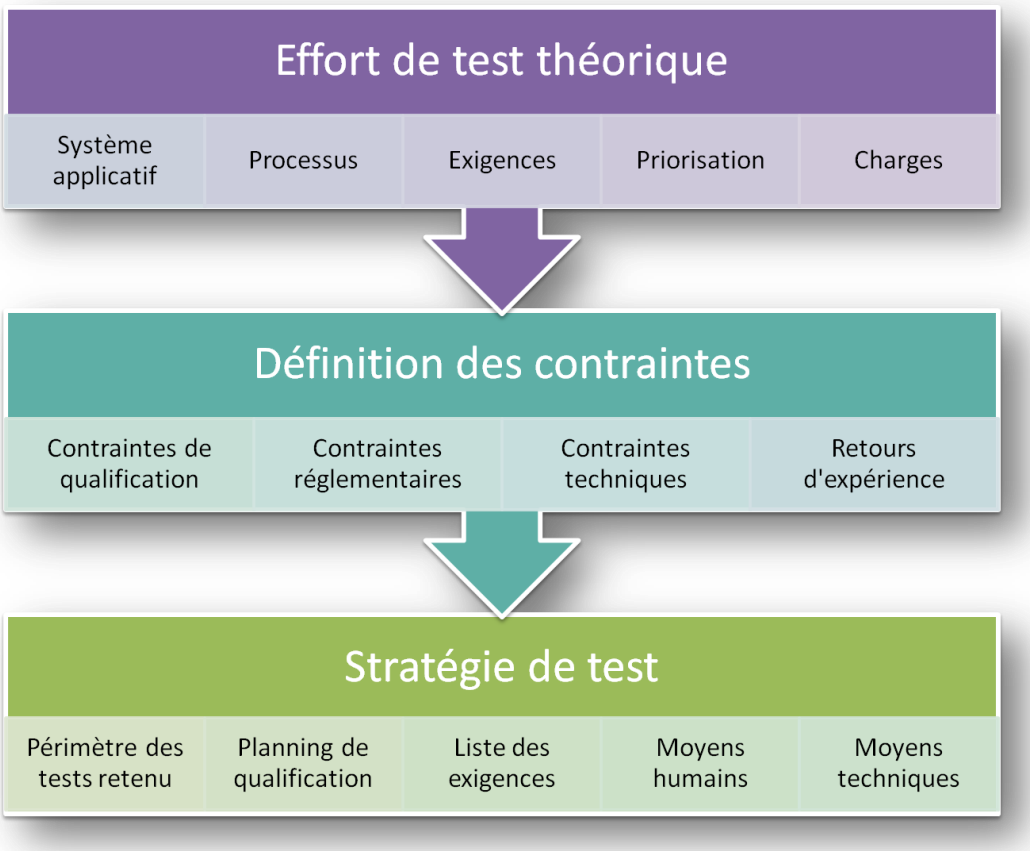
  Le Chef de Projet conçoit le périmètre des tests à partir des documents à sa disposition : expressions de besoins, spécifications techniques et fonctionnelles etc. Il établit ensuite un planning sans tenir compte des spécificités du client.

- **La stratégie de test réelle :**

  Le Chef de Projet prend en compte l'environnement du client avec ses limitations et ses spécificités et applique ces contraintes tant au périmètre des tests théoriques qu'au planning initial. Une fois validée par tous les intervenants au projet, la stratégie va servir de référentiel tout au long de la phase de qualification.

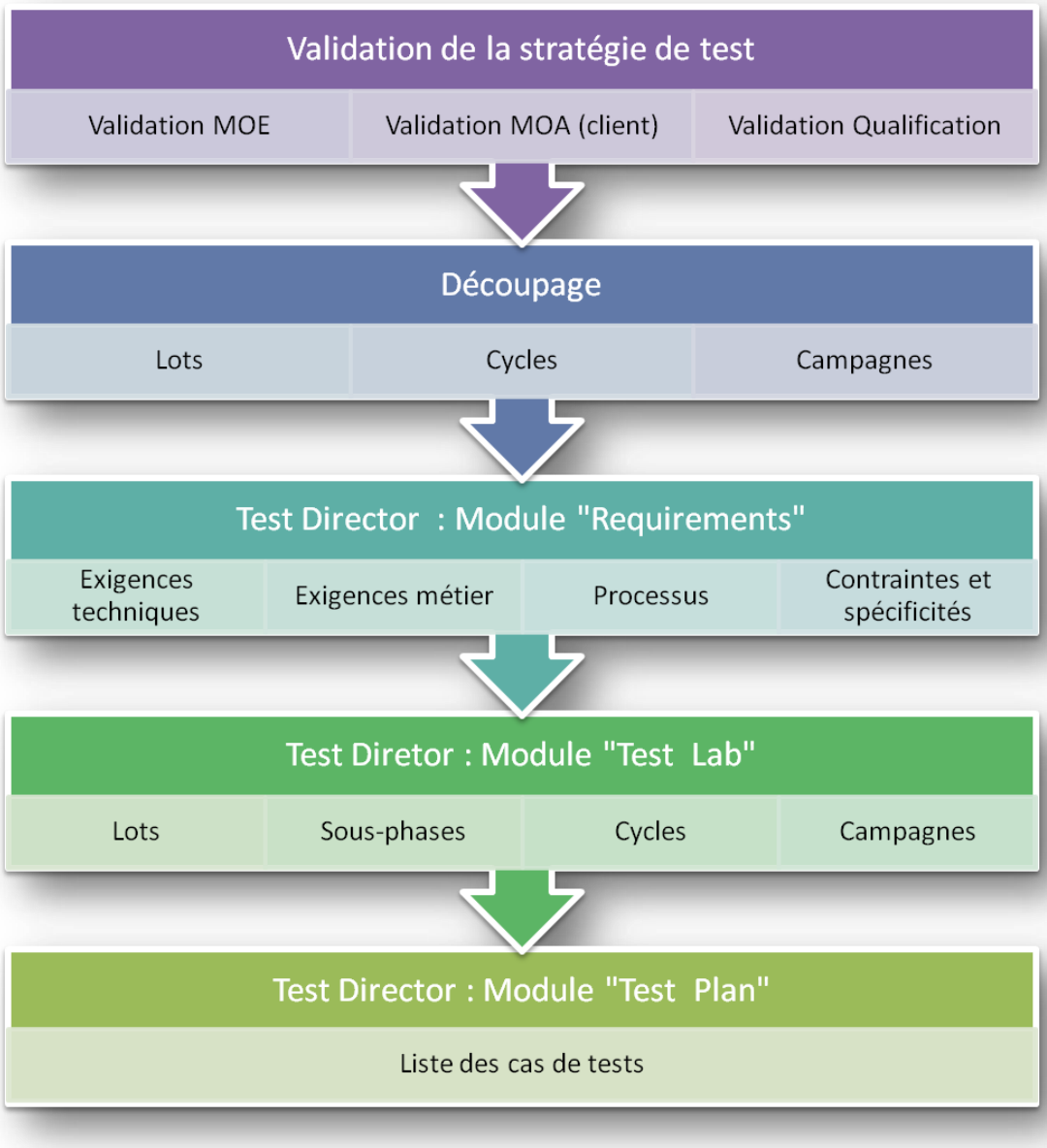
  Plus précisément, l'établissement d'une stratégie de test doit suivre le processus ci-dessous :




Effort de test théorique
Système applicatif
Processus
Exigences
Priorisation
Charges
Définition des contraintes
Contraintes de qualification
Contraintes réglementaires
Contraintes techniques
Retours d'expérience
Stratégie de test
Périmètre des tests retenu
Planning de qualification
Liste des exigences
Moyens humains
Moyens techniques

2. Modules de Test Director

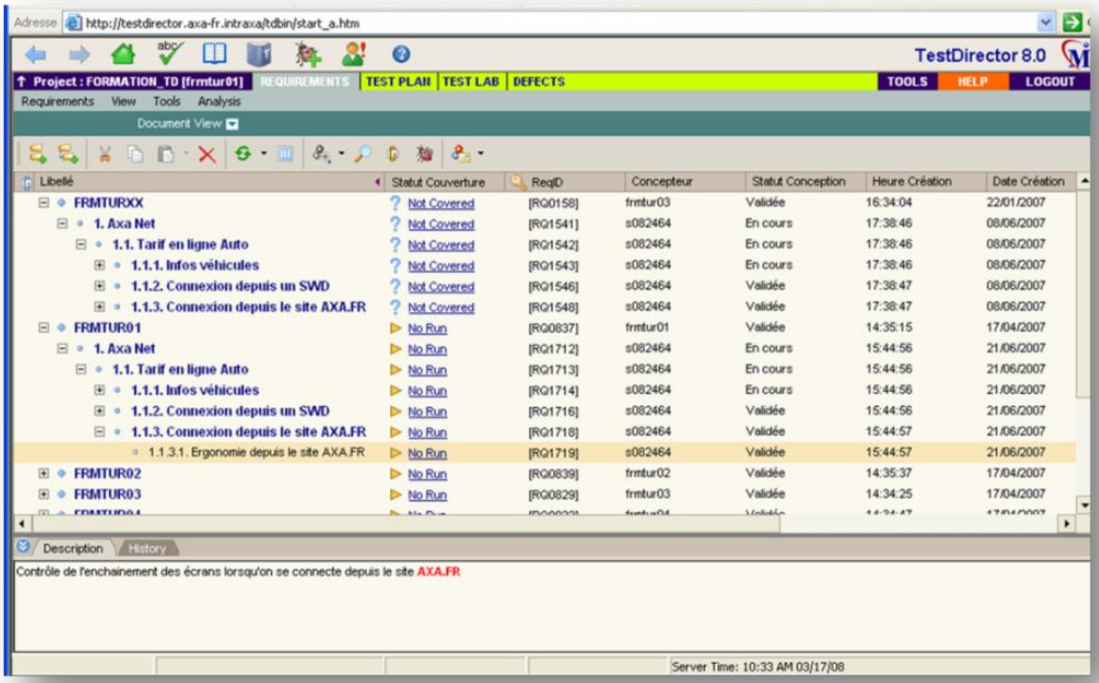
Une fois la stratégie définie, le Chef de Projet a pour tâche de découper la phase de qualification en chantiers, cycles et campagne de tests. Il peut dès alors intégrer les informations au sein de Test Director :



Ainsi Test Director organise et suit tout le cycle de la recette à travers des onglets distincts :

2.1 Onglet « Requirements »

C'est au sein de cet onglet que le Chef de Projet définit la liste des exigences. Chacune d'elle doit être complétée par une priorité et pointer vers le document source.



Les colonnes principales sont :

- « **Libellé** » : On y trouve l'arborescence des dossiers où le dernier niveau correspond à l'exigence elle-même. Il est généralement conseillé de ne pas dépasser 4 niveaux.
- « **Statut couverture** » : Cette colonne permet de vérifier si l'exigence a été couverte par au moins un cas de test et indique dans ce cas l'état de son exécution.
- « **ReqID** » : Il s'agit d'un identifiant unique pour chaque exigence.
- « **Concepteur** » : Nom ou login de la personne qui intégré l'exigence au sein de test Director.
- « **Statut Conception** » : Le Chef de Projet indique ici si l'exigence a été validée. Si tel n'est pas le cas, le Chargé de Qualification ne sera pas en mesure d'exécuter les cas de tests associés.
- « **Priorité** » : Le Chef de Projet classifie le niveau de priorité à l'exigence.
- « **Type** » : Cette colonne permet d'organiser les exigences selon leur origine : technique, fonctionnelle …

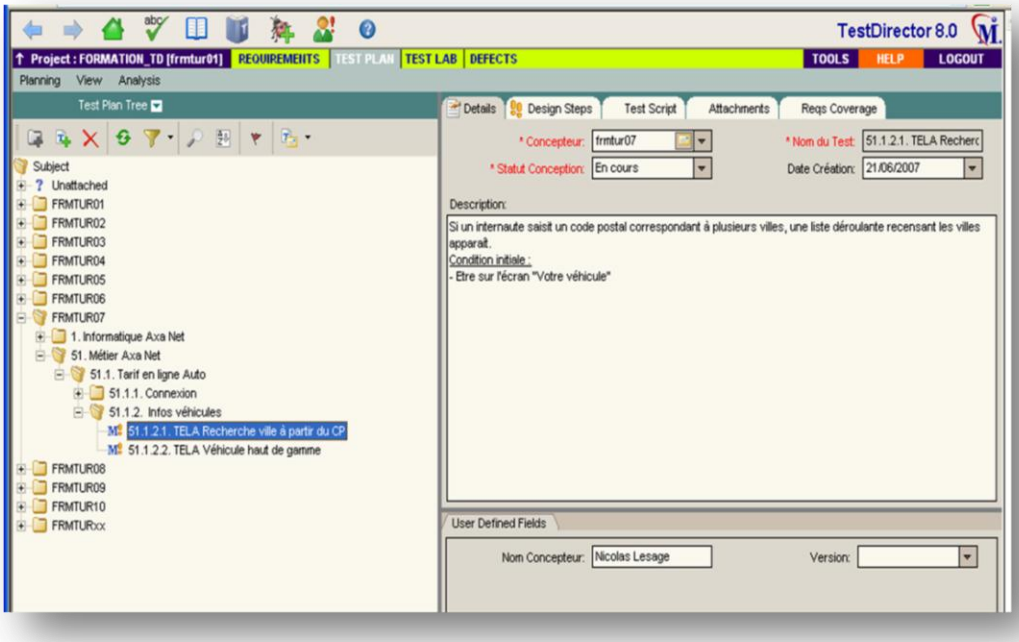
2.2 Onglet « Test Plan »

Une stratégie de test bien organisée offre au Chargé de Qualification une base solide pour concevoir le plan de test.

Tout d'abord, il définit l'arborescence du plan. Il s'appuie généralement sur les fonctionnalités du produit à tester.

Puis il créé les cas de tests et les étapes en fournissant une description explicite pour le testeur (qui peut en effet être une personne différente).

Enfin, il associe chaque cas de test à une ou plusieurs exigences afin de garantir la traçabilité de son travail.

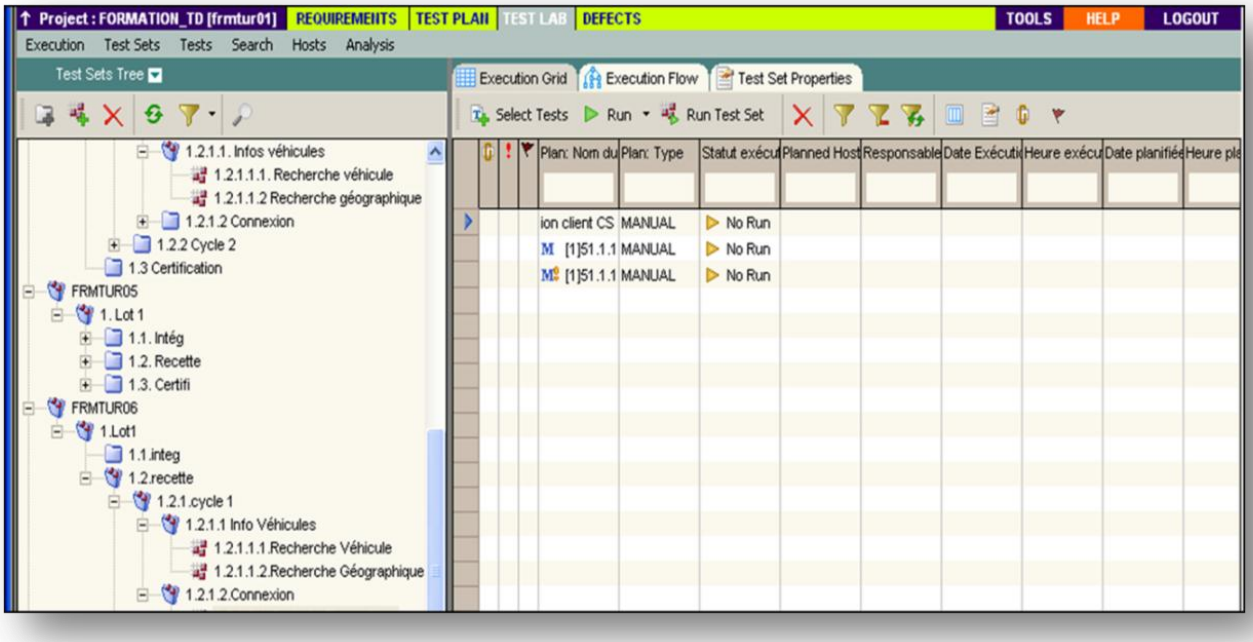


L'onglet « Test Plan » est découpé en deux fenêtres distinctes :

- La fenêtre de gauche contient l'arborescence du plan de test où le dernier niveau correspond au cas de test lui-même.
- La fenêtre de droite détaille les informations propres au cas de test :
  - « **Détails** » : Cet espace permet de donner des explications sur le cas de tests à exécuter. Il indique également qui a créé le cas de tests, son statut ou encore sa date de conception.
  - « **Design Steps** » : Ici sont listées les étapes du cas du cas de test
  - « **Reqs Coverage** » : C'est ici que le cas de test est associé à une ou plusieurs exigences.
  - « **Test Script** » : Cette fenêtre offre la possibilité de joindre un script que le testeur devra exécuter. Cette fonctionnalité est très peu employée.
  - « **Attachments** » : On y stocke ici des fichiers qui peuvent présenter une importance pour la bonne exécution du cas de test : un aperçu d'écran, par exemple.

2.3 Onglet « Test Lab »

Cet onglet offre la possibilité d'organiser les scénarii en lots, cycles et campagnes. Au niveau le plus bas, on y retrouve donc le scénario lui-même avec, sur une fenêtre dédié, la liste des tests à exécuter issus du plan de test.



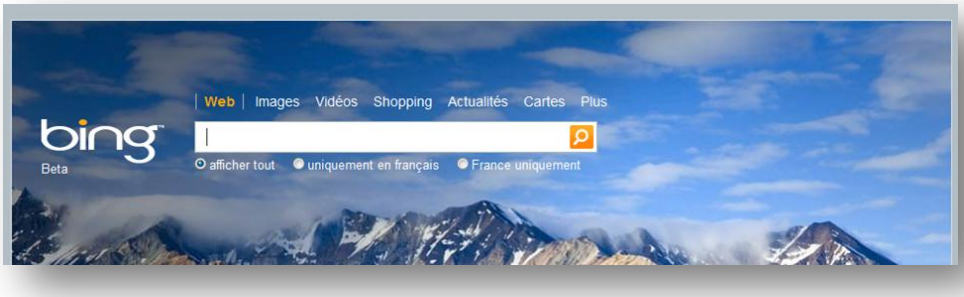
Cette fenêtre de droite constitue donc l'interface principale pour le testeur.

Elle est constituée des onglets suivants :

- « **Execution Grid** » : On y trouve la liste des cas de tests à exécuter pour le scénario sélectionné.
- « **Execution Flow** » : s'affiche ici une représentation schématique de type 'Workflow' de l'enchainement des scénarii. Cette fonctionnalité est très peu utilisée.
- « **Test Set Properties** » : Cet onglet permet de saisir des informations complémentaires sur le scénario : détails, nom du concepteur etc.

2.4 Distinction entre plan de test et scénarios

Pour bien comprendre la distinction entre plan de test et scénario, il est intéressant de partir d'un exemple concret : imaginons, par exemple, la recette du moteur de recherche « Bing » :



Il est possible de découper cette interface de la manière suivante :

- **Onglet** : Accès à la page d'accueil
- **Pavés** : Accès aux fonctionnalités :
  - Web
  - Images
  - Shopping
  - Actualités
  - Cartes
  - Plus
- **Champ** : Insertion de l'information à rechercher selon le critère sélectionné au niveau « pavé »

La rédaction des campagnes au sein du « Test Lab » de Test Director peuvent suivre la logique suivante :

- **Test de surface :**

  L'application est testée à un niveau basique afin de vérifier sa stabilité et la cohérence générale des différentes fonctionnalités.

  On vérifie ici que l'URL est accessible et la page s'affiche quelque soit le navigateur utilisé. On peut également tester le retour fourni par le moteur en entrant des informations divers pour chacun des pavés.

- **Test bilatéral :**

  Les différentes fonctionnalités de l'application sont testées plus en profondeur. Tout d'abord, on vérifie la cohérence des résultats observés : ce sont les tests positifs. A

l'inverse, on teste la réaction de l'application si les éléments en entrée ne correspondent pas à ce qui est prévu : on parle alors de tests négatifs.

On vérifie par exemple qu'en entrant le mot « maison », la réponse au niveau du pavé « Web » est effectivement une liste de sites consacrés à ce sujet. Une recherche de ce même avec la fonction « Images » affichera alors une succession de photos ou de dessins correspondant au mot entré. Pour les tests négatifs, on peut vérifier la réaction du moteur lorsque l'on entre une chaîne de caractères telle que « #* ».

- **Test avancé :**

  L'intégralité de l'application est passée en revue. Chaque fonctionnalité est testée dans ses moindres détails. A ce niveau, les aspects techniques sont également traités.

  Des exemples explicites sont utilisés pour vérifier les réponses fournies par le moteur de recherche et, cela, pour tous les pavés et pour toutes les options. En termes techniques, on testera par exemple les temps de réponse, les mots longs, la charte graphique etc.

- **Test de non régression :**

  On vérifie à ce niveau que l'application testée ne modifie en rien le comportement des autres applications avec lesquels elle cohabite.

  Dans le cas du moteur de recherche, on peut vérifier que l'application « Cartes » accessible depuis le pavé correspondant n'a subit aucune dégradation.

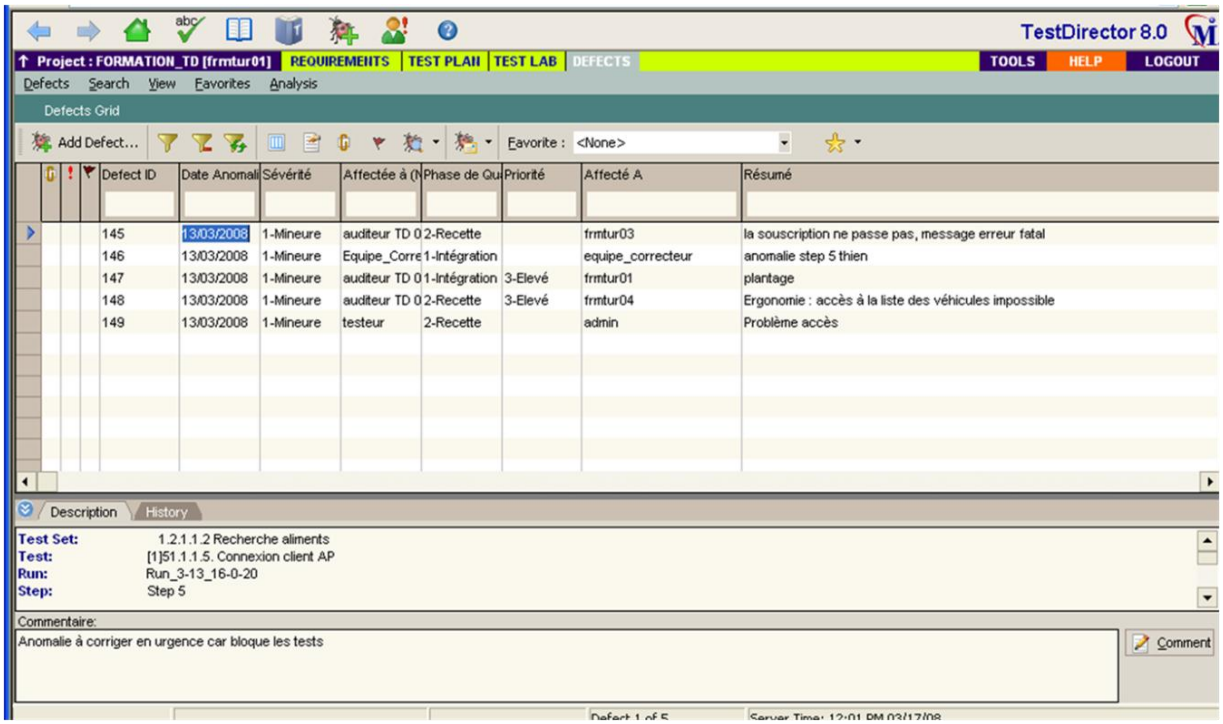
2.5 Onglet « Defects »

A l'exécution du scénario, le testeur a pour objectif de détecteur le moindre écart par rapport au résultat attendu. Il déclare ainsi une anomalie dans Test Director, décrit le problème rencontré et lie cette anomalie au cas de tests en cours.

Le Chef de Projet Qualification prend alors la main, analyse la pertinence de cette anomalie, puis l'envoie à l'équipe concernée.

Cette anomalie est alors corrigée par l'équipe responsable et édite un document expliquant les détails de la résolution.

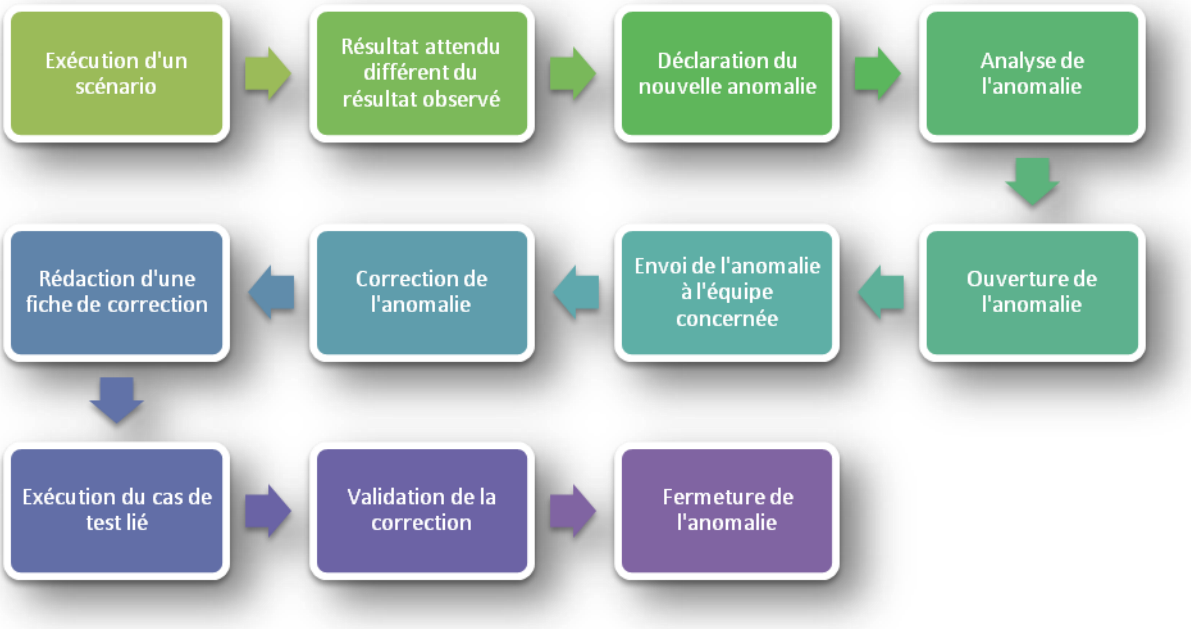
A la réception du document, le Chef de Projet autorise le testeur à relancer l'exécution du cas test associé. Si le résultat attendu est cohérent, l'anomalie est alors déclarée comme « corrigée ».



Au sein de Test Director, l'anomalie est identifiée principalement par les colonnes suivantes :

- « **DefectID** » : C'est un didentifiant unique pour l'anomalie.
- « **Date Anomalie** » : Date à laquelle l'anomalie a été déclarée.
- « **Sévérité** » : Niveau de sévérité de l'anomalie. Cette information est initialisée par le testeur lors de la création de l'anomalie puis modifiée si nécessaire par le Chef de Projet lors de son analyse.
- « **Affectée à** » : Nom de la personne ou de l'équipe en charge de l'anomalie. Cette donnée est généralement du Chef de Projet qui affecte l'anomalie selon son origine : métier, technique …
- « **Phase de Qualification** » : Un même cas de test peut être utilisé dans plusieurs phases de la recette. Il est donc intéressant de savoir à laquelle l'anomalie est associée.
- « **Résumé** » : Brève description de l'anomalie.

En résumé, le cycle de vie d'une anomalie suit le cheminement suivant :

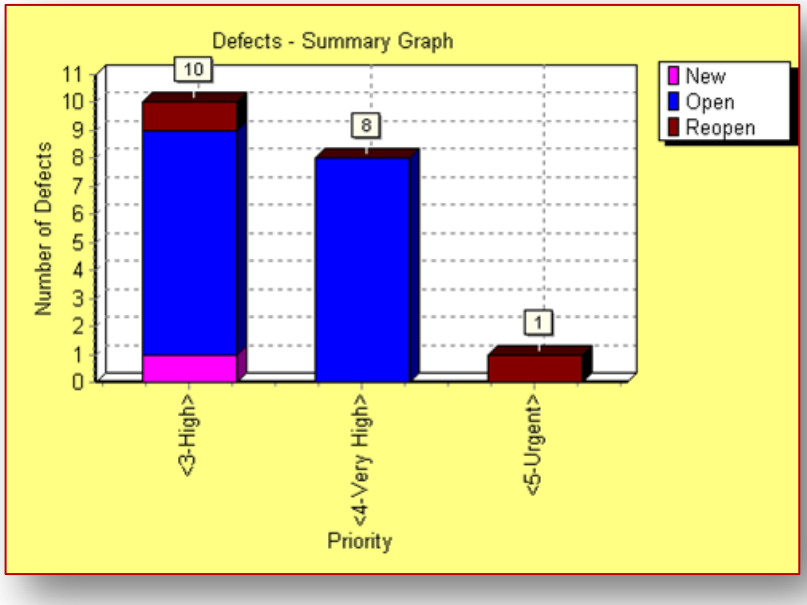


2.6 Onglet « Stats »

Il est important pour le Chef de Projet d'avoir une vision synthétique du projet à un instant donné, notamment pour la préparation de compte-rendu ou de comités de pilotage.

Test Director met à disposition un outil de création de statistiques. Ainsi, il est possible de créer un graphique à barres à partir de la situation du plan de test, de la campagne ou encore des anomalies.

Par exemple, il peut être utile de connaître l'état des anomalies en fonction de leur criticité. On aurait un graphique qui ressemblerait à celui-ci :


Defects - Summary Graph
Number of Defects
0
1
2
3
4
5
6
7
8
9
10
11
10
8
1
New
Open
Reopen
<3-High>
<4-Very High>
<5-Urgent>
Priority

IX INSTALLATION ET INDUSTRIALISATION

L'installation est une étape à ne pas négliger. Elle doit se dérouler de la manière la plus claire avec l'appui d'une documentation explicite et sous l'égide de personnes compétentes. Il s'agit non seulement de maîtriser la première installation sur un environnement vierge mais également les réinstallations suivantes, notamment à la suite d'une correction ou d'une mise à jour du produit.

On pourra alors parler d'industrialisation lorsque ces deux process, installation et réinstallation, se déroulement parfaitement.

1. Installation

C'est au Chef de Projet Qualification de donner le feu vert pour les premiers tests d'installation : toutes les anomalies de recette doivent être corrigées et closes, les tests de bout en bout satisfaisants.

Les composants à installer doivent être identifiés dans la gestion de configuration. La procédure d'installation doit également distinguer les opérations manuelles des tâches automatiques.

Un cahier d'installation doit afficher une vision temporelle de la procédure avec des estimations de durée pour chaque étape. Ce cahier devrait en outre contenir :

- Les pré-requis matériels (configuration minimale, recommandée …),
- Les pré-requis logiciels,
- Les ressources humaines nécessaires,
- La liste des composants à installer,
- Les procédures automatiques,
- Les procédures manuelles,
- Un organigramme de planification de l'installation,
- Un mode d'emploi.

La rédaction et l'exécution des premiers tests d'installation doit réunir :

- La MOE,
- L'exploitation,
- Et optionnellement, la qualification pour un appui en termes d'organisation des tests.

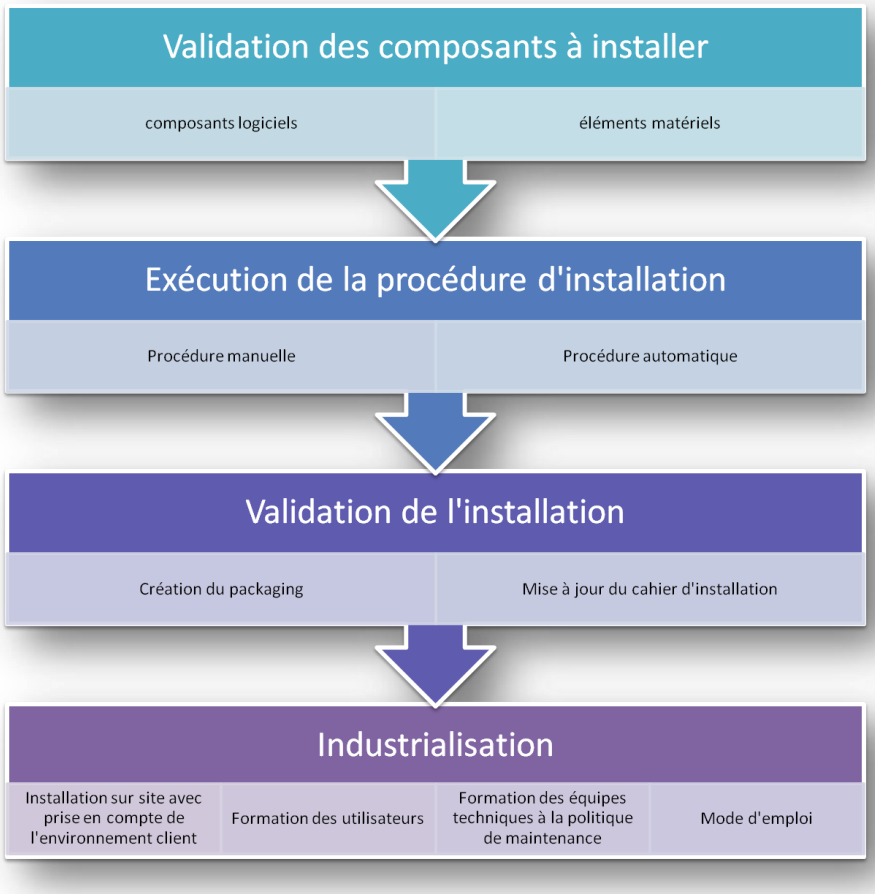
2. Industrialisation

L'industrialisation, c'est à dire le déploiement de l'application en production, doit prendre en compte les spécificités des sites clients. En outre, si le produit développé est destiné à remplacer un logiciel déjà présent chez le client, il faudra prévoir un calendrier de déploiement avec l'équipe d'exploitation. Il est nécessaire, dans ce cas, d'assurer les activités du site avec un fonctionnement en parallèle de manière à pouvoir effectuer un retour en arrière si une défaillance ou une incompatibilité se produit sur le nouveau progiciel.

Cette phase d'industrialisation oblige le fournisseur à :

- Définir une stratégie de communication pour être le plus réactif possible en cas de d'anomalie,
- Prendre en charge ces anomalies, de leur analyse à leur correction,
- Fournir une documentation claire et explicite adapté au public visé : équipe métier, techniciens, utilisateurs finaux …
- Offrir une formation à l'utilisation mais également à la maintenance du produit.

Le schéma suivant décrit les étapes nécessaires à mener lors de la première installation jusqu'au déploiement final.



3. Exploitation

Lorsque le produit a été correctement implémenté sur le site client, le fournisseur peut désormais passer la main à l'équipe d'exploitation.

Cette équipe va procéder, selon l'ampleur du projet, à une montée en charge progressive et une surveillance accrue sur l'interaction du produit livré avec l'environnement existant. Elle pourra requérir l'aide du fournisseur pendant une période de garantie définie contractuellement.