**PROJET : Méthode Guillarme**

**Document**

**Plan de Tests**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document :**  **Plan de Tests** | | | |
| **Référence** : FOR- Guillarme-2024-06 | | **Auteurs :** Clara VESVAL | |
| **Service :** Ynov | **Etat :** Préliminaire | | **Type de diffusion :** Restreinte |

HISTORIQUE DES REVISIONS

|  |  |  |
| --- | --- | --- |
|  | **Rédacteurs** | **Vérificateur** |
| **Nom** | Clara VESVAL | Rémi BOURDILLON |
| **Date** | 27/06/2024 |  |

|  |  |  |
| --- | --- | --- |
| **Version** | **Description de la modification - Auteurs** | **Date** |
| **1.0** | Mise en place du document, architecture, design | 27/06/2024 |
| 1.1 | Début du PdT de l’application « Méthode Guillarme » |  |
| 1.2 | Focus et Analyse sur la partie algorithmique de la gestion des séries dans « Méthode Guillarme » |  |
| 1.3 | Choix de l’environnement et de la stratégie à mettre en place |  |
| 1.4 | Changement d’envrionnement et de stratégie pour plus de portablité, automatisation et de visibilité des résultats pour faciliter l’analyse des bugs |  |
| 1.5 | Ecriture des cas Nominaux |  |
| 1.6 | Ajout de tests pour prendre en compte le nombre de vie |  |
| 1.7 | Tests de robustesse sur les données du csv (données manquantes, ou corrompues, etc) |  |
| 1.8 | Ajout de tests sur la réinitialisation d’une série |  |
| 1.9 | Ajout de test à la suite d’un bug : Enlève une vie lors d’une pratique incomplète alors qu’il y avait une autre séance plus tard dans la même journée |  |
| 1.10 | Ajout tests si plusieurs timestamps identique pour un même SessionId |  |
| 1.11 | Ajout Tests vie regagnée |  |
| 1.12 | Ajout tests de performances pour gérer de gros fichier csv |  |

SOMMAIRE

[TABLEAUX 5](#_Toc171864332)

[FIGURES 5](#_Toc171864333)

[1. Introduction 5](#_Toc171864334)

[Objet du document 5](#_Toc171864335)

[Domaine d’application 6](#_Toc171864336)

[Présentation du projet 6](#_Toc171864337)

[Règle métier 6](#_Toc171864338)

[Spécification Technique d’Interface (STI) 7](#_Toc171864339)

[Hypothèses et risques 8](#_Toc171864340)

[Anomalies Critiques Bloquantes 8](#_Toc171864341)

[Complexité de Résolution 8](#_Toc171864342)

[Non-Résolution d'Anomalies 9](#_Toc171864343)

[Abandon d'une fonctionnalité 9](#_Toc171864344)

[Impact sur les Délais 9](#_Toc171864345)

[Communication et Collaboration 9](#_Toc171864346)

[Gestion des Risques 9](#_Toc171864347)

[Documentation et Suivi 9](#_Toc171864348)

[2. Terminologie 10](#_Toc171864349)

[Abréviations 10](#_Toc171864350)

[Glossaire : Définitions des termes employés 10](#_Toc171864351)

[Documents applicables 11](#_Toc171864352)

[3. Organisation de la validation 12](#_Toc171864353)

[Rôles et responsabilités 12](#_Toc171864354)

[Documents de tests 12](#_Toc171864355)

[Environnement de test 12](#_Toc171864356)

[Environnement Matériel 13](#_Toc171864357)

[Environnement Logiciel 13](#_Toc171864358)

[4. Ressources 14](#_Toc171864359)

[Ressources humaines 14](#_Toc171864360)

[Ressources matérielles 14](#_Toc171864361)

[5. Stratégie de test 14](#_Toc171864362)

[Stratégie de validation 15](#_Toc171864363)

[Principe 15](#_Toc171864364)

[Organisation de la stratégie 15](#_Toc171864365)

[Objectif de test 17](#_Toc171864366)

[Niveaux de criticité 18](#_Toc171864367)

[Critères d’arrêt des tests 18](#_Toc171864368)

[Circuit des modifications 18](#_Toc171864369)

[6. Description des tests 20](#_Toc171864370)

[Le cycle de développement 20](#_Toc171864371)

[Validation des projets 21](#_Toc171864372)

[Dossier de tests unitaires 21](#_Toc171864373)

[Dossier de tests d’intégration 21](#_Toc171864374)

[Dossier de tests de validation 21](#_Toc171864375)

TABLEAUX

[Tableau 2‑1 : Abréviations 12](#_Toc3302362)

[Tableau 2‑2 : Glossaire 13](#_Toc3302363)

[Tableau 2‑3 : Documents Applicables 13](#_Toc3302364)

[Tableau 4‑1 : Ressources matérielles 18](#_Toc3302369)

[Tableau 5‑1 : Organisation de la stratégie 20](#_Toc3302362)

FIGURES

[Figure 5‑1 : Cycle de prise en compte 23](#_Toc3302370)

[Figure 6‑1 : Cycle de développement logiciel en vigueur 24](#_Toc3302371)

# Introduction

## Objet du document

Ce document a pour objectif de décrire la méthodologie concernant le test et la validation d’application. Il aborde successivement l'organisation, la planification des tests et la préparation des campagnes de tests à effectuer.

Il aborde les points suivants :

* L’organisation du projet de test validation, rôles et responsabilités des intervenants dans le processus de test ;
* Les activités de préparation et de conduite des tests ;
* La stratégie de test de validation appliquée au projet ;
* Les outils et les environnements dans lesquels les tests doivent s'exécuter ;
* Les niveaux de criticité et les critères d'arrêt des tests ;
* Le cycle de prise en compte des anomalies rencontrées dans l'exécution des tests ainsi que leur traitement.

Ce document est applicable sur les modules entrant dans le cadre du développement de l'application à développer et doit couvrir tous les scénarios possibles, y compris les cas limites et les situations inhabituelles.

## Domaine d’application

Ce document est applicable l’algorithme de la gestion des séries dans « Méthode Guillarme », pour incrémenter la série de l’utilisateur lorsque qu’il pratique est un couple séance assis et séance allongé le même jour *(Détail cf Présentation du projet / Règle métier)*

## Présentation du projet

La Méthode Guillarme est une rééducation basée sur le souffle permettant de retrouver un abdomen et un diaphragme compétents au service de votre périnée, de votre posture, de votre système digestif, de votre esthétique.

L'objectif est d'entraîner le ventre à orienter les pressions vers le haut lors des efforts quotidiens.

Pour fidéliser l’utilisateur, un système de suivi est mis en place pour incrémenter sa série de jour d’affilé pratiqué, ainsi qu’une gestion de vie pour compenser l’absence de pratique.

### Règle métier

Un exercice avoir une durée de 5 min (niveau=1) ou de 10 min (niveau=2), et se faire soit assis ou allongé

Un utilisateur doit effectuer 2 exercices de 5 min ou 1 exercice de 10 min pour compléter une séance.

Une pratique est un couple séance assis et séance allongé le même jour.

Chaque jour d’affilé pratiqué incrémente la série de +1.

L’utilisateur dispose de 2 vies pour pallier l’absence de pratique sur un jour

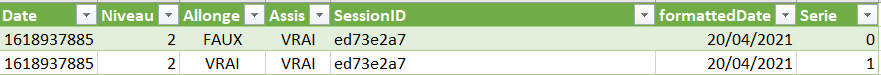
Chaque jour sans pratique consomme 1 vie.

Lorsque l’utilisateur a consommé plus de vie qu’il n’en avait

* La série est « cassée » et reprend de 0
* 2 vies sont recréditées

### Spécification Technique d’Interface (STI)

Les données d’entrée de l’algorithme sont au format csv suivant :

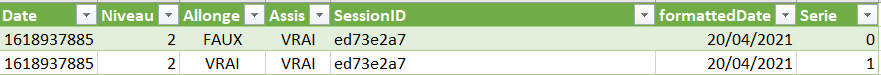


|  |
| --- |
| **Date,"Niveau","Allonge","Assis","SessionID","formattedDate"** |
| 1618937885,"2","False","True","ed73e2a7 ","20/04/2021" |
| 1618937885,"2","True","True","ed73e2a7 ","20/04/2021" |

* **Date :** Timestamps 🡪 Date avec l’heure indiquant quand a été réalisé l’exercice
* **Niveau**: Integer (1 ou 2) ou vide 🡪 Valeur de l’exercice réalisé de 5 min (niveau=1) ou de 10 min (niveau=2), ou pas de pratique (vide)
* **Allonge :** Boolean (True/False) 🡪 Si l’exercice a été fait Allongé (True) ou pas (False)
* **Assis :** Boolean (True/False) 🡪 Si l’exercice a été fait Assis (True) ou pas (False)
* **SessionID :** String 🡪 Identifiant de l’utilisateur
* **formattedDate :** Date (JJ/MM/AAAA) 🡪 Date formaté indiquant quand a été réalisé l’exercice

Selon les règles métiers, il faut ajouter le nombre de série pour chaque ligne, le reste des données doit rester identiques à celui d’entrée

Les données de sortie de l’algorithme sont au format csv suivant :



|  |
| --- |
| **Date,"Niveau","Allonge","Assis","SessionID","formattedDate", Serie** |
| 1618937885,"2","False","True","ed73e2a7 ","20/04/2021", 0 |
| 1618937885,"2","True","True","ed73e2a7 ","20/04/2021", 1 |

* **Serie :** Integer >= 0 🡪 Nombre de jour d’affilé pratiqué

## Hypothèses et risques

Chaque application comporte des hypothèses sur la fiabilité et sur le risque de glissement du projet. Il s’agit de définir l’ensemble des incidents pouvant survenir pendant la phase de test.

### Anomalies Critiques Bloquantes

La rencontre d'anomalies critiques bloquantes, telles que des plantages d'application ou des erreurs de sécurité majeures, peut retarder le développement.

Par exemple :

* **La variance liée à la « Culture »**

Tout d’abord si le format csv n’est pas comme prévu, il peut subvenir des problèmes durant le traitement, et/ou à la sortie du fichier. De plus, le format des dates/timestamps peuvent également poser problèmes et enfin l’ncodage des caractères UTF-8 vs ISO

* **La variance liée à l’OS**

L’algorithme doit pouvoir être portable sur tous les environnements

* **Problèmes de Performance :**

Le programme peut être lente à répondre, en particulier lors du chargement de grandes quantités de données et se perdre dans les nombreuses lignes du csv, ou pas pouvoir toutes les réécrire.

### Complexité de Résolution

La complexité de résolution des anomalies peut entraîner des retards supplémentaires sur le projet, en particulier si des ajustements importants dans le code sont nécessaires.

### Non-Résolution d'Anomalies

En cas de non-résolution d'une anomalie critique lors des tests d'intégration, l'item concerné devra être revu par l’ensemble de l’équipe, pour prendre une décision. Ne pas l’ajouter à l'application, afin de garantir la stabilité du produit final, la repenser entièrement ou demande une aide extérieure par exemple.

### Abandon d'une fonctionnalité

L'équipe peut décider d'abandonner un item si la résolution de ses anomalies implique plus de perte de temps que ce qu'elle apporte fonctionnellement à l'application. Cela peut se produire lorsque les fonctionnalités sont jugées non essentielles ou peu utilisées par les utilisateurs finaux.

### Impact sur les Délais

Les retards dus à la résolution d'anomalies critiques peuvent avoir un impact sur les délais du projet, en particulier si ces anomalies sont découvertes tardivement dans le cycle de développement. C’est pour cela que les tests tout au long du développement sont essentiels.

### Communication et Collaboration

Une communication efficace et une collaboration étroite entre les membres de l'équipe de développement, les testeurs et les parties prenantes sont essentielles pour identifier rapidement et résoudre les anomalies.

### Gestion des Risques

Une évaluation continue des risques tout au long du projet est nécessaire pour anticiper les problèmes potentiels et prendre des mesures préventives appropriées comme la défaillance de service tiers qui peuvent rencontrer des problèmes de disponibilité ou de performance. (Un document a été réalisé au préalable du développement)

### Documentation et Suivi

La documentation précise des anomalies rencontrées, de leur résolution et des décisions prises concernant les items abandonnés est cruciale pour assurer la traçabilité et la transparence dans le processus de développement.

En tenant compte de ces hypothèses et en adoptant des pratiques de gestion des risques efficaces, l'équipe de développement peut atténuer les impacts négatifs des incidents survenant pendant la phase de test et maintenir le projet sur la bonne voie et s'assurer que l'application est prête pour un déploiement réussi.

# Terminologie

## Abréviations

|  |  |
| --- | --- |
| DTV | **D**ossier de **T**est de **V**alidation |
| PTV | **P**lan de **T**est de **V**alidation |
| T&V | **T**est et **V**alidation |
| PdT | **P**lan de **T**est |
| PMOT | **P**lan de **M**ise en **Œ**uvre des **T**ests |

Tableau 2‑1 : Abréviations

## Glossaire : Définitions des termes employés

|  |  |
| --- | --- |
| **Campagne de Test** | Activité qui consiste à dérouler un ensemble de jeux de test. Un dossier de test est produit à l’issue d’une campagne. |
| **Cas de Test** | Déclinaison d’un test précisant les valeurs utilisées pour les variables du test ainsi que les résultats attendus. |
| **Dossier de Test** | Ensemble documentaire qui contient la description des scénarios et cas de tests, ainsi que l’exécution des jeux de test. Le dossier de test est le reflet d’une campagne de test. |
| **Jeux de Test** | Ensemble de scénarios et cas de tests permettant de tester un produit logiciel. L’entrainement des cas et scénarios de tests est relatif à une stratégie de tests précisée dans le plan de test. |
| **Plan de Test (ou Protocole de Recette)** | Document décrivant le déroulement d'un jeu de test : Stratégie de test, critères d'arrêt, planification. |
| **Scénarios de Tests** | Ensemble de cas de tests cohérents permettant de traiter un objectif fonctionnel. |
| **Test Fonctionnel** | Test (vu de l'utilisateur) du bon fonctionnement d'un produit logiciel, d'une fonctionnalité ou d'une fonction de base. Vérification par rapport aux spécifications. |
| **Test de Non-Régression** | Vérification qu'une nouvelle version du produit logiciel fonctionne sans dégradation (technique, fonctionnelle, performance) par rapport à la version précédente. |
| **Test Unitaire** | Vérification qu'un module, objet, ou autres composants d'un produit logiciel est conforme à ses spécifications (techniques et fonctionnelles). Basé sur des tests fonctionnels (vue de l'extérieur) et structurels (vue de l'intérieur). |
| **Test ISO périmètre** | Vérification que la nouvelle application développée réalise et couvre à l’identique l’ensemble des fonctionnalités présente dans l’ancienne application. |
| **Test de Validation** | Vérification que le produit logiciel est cohérent et complet par rapport aux spécifications fonctionnelles. |

Tableau 2‑2 : Glossaire

## Documents applicables

|  |  |
| --- | --- |
| **Identification** | **Description** |
| [REF 1] | Support de cours : C\_-\_Jeux\_3D\_OpenGL\_-\_Tests.pdf de C.Vidal |
| [REF 2] | Support de cours : 04-StrategieTests.doc de C.Vidal |
| [REF 3] | Support de cours : Academy-MethodoTest-Cours 4.pdf de R.Bourdillon |
| [REF 4] | Support de cours : Academy-MethodoTest-Cours 5.pdf de R.Bourdillon |
| [REF 5] | Support de cours : Exemple de PdT/PMOT.xls de R.Bourdillon |

**Tableau 2‑3 : Documents applicables**

# Organisation de la validation

## Rôles et responsabilités

La réalisation et analyse des tests est effectuées par la chef de projet et le responsable. Les tests unitaires sont réalisés par les programmeurs au moment du développement. Les tests d’intégration sont effectués par l’équipe.

Le traitement des anomalies est, autant que possible, à la charge de la personne ayant développées la fonctionnalité à l’origine de l’anomalie, ou la personne disponible au moment de la découverte selon l’urgence de la correction nécessaire.

## Documents de tests

Les documents suivants sont générés par l’équipe et constituent le référentiel documentaire de test. Ils sont à nouveau appliqués dans les tests de non-régression lors de la livraison de nouvelles versions de l’application.

Ces documents incluent :

* La stratégie de tests / Plan de Tests
* Les fiches de tests / Plan de Mise en Œuvre des Tests

Les documents sont édités à l’aide des outils bureautiques tels que Libre Office ou Word. Les données de tests générées par les activités sont identifiées par :

* Copie de la version testée de l’application ;
* Fiches d’anomalies éventuellement accompagnée de capture d’écran.

Tous ces éléments doivent pouvoir être identifiés et liés à chaque version du logiciel.

## Environnement de test

On définit l’environnement de test devant être mis en œuvre : matériel et logiciel. La description précise de l’environnement est spécifiée pour l’ensemble de la campagne de test.

### Environnement Matériel

**Ordinateurs de Développement**

Ordinateurs de bureau ou portables avec des spécifications suffisamment robustes pour exécuter les environnements de développement et les outils de test de manière fluide :

* Résolution d'écran minimale de 1 280 x 800
* 8 Go de RAM ou plus
* Au moins 8 Go d'espace disque disponible (IDE)
* OS
  + Linux
    - Toute distribution Linux 64 bits compatible avec Gnome, Unity KDE. GNU C Library (glibc) 2.31 ou version ultérieure
    - Architecture de processeur x86\_64, processeur Intel Core (2e génération ou version ultérieure), ou AMD processeur AMD compatible avec la virtualisation AMD (AMD-V) et SSSE3.
  + MacOS.
    - MacOS® 10.14 (Mojave) ou version ultérieure
    - Puces ARM ou Intel Core de 2e génération ou version ultérieure, compatibles avec Hypervisor Framework
  + Windows
    - Microsoft® Windows® 8/10/11, version 64 bits
    - Architecture de processeur x86\_64, Intel Core (2e génération ou version ultérieure), ou processeur AMD compatible avec Windows Hypervisor

### Environnement Logiciel

**Obligatoire :**

* Un terminal
* Un navigateur Web

**Optionnel :**

* Python >= 3.X
  + Pandas
  + Pyinstaller
  + BeautifulSoup

# Ressources

## Ressources humaines

L’équipe de projet assure les tâches de préparation, d’élaboration et de réalisation des tests. Elles composent également l’équipe de développement et sont responsables du traitement des anomalies.

L’effort alloué à la résolution des problèmes rencontrés lors de l’exécution des scénarios comprend :

* L’analyse du problème par l’équipe ;
* La détermination des modifications à apporter par le chef de projet et/ou le responsable qualité.

## Ressources matérielles

Les ressources matérielles nécessaires à l’accomplissement des tâches sont définies dans la tableau suivant :

|  |  |  |
| --- | --- | --- |
| Tâche | Matériel | Observations |
| Préparer le Plan de Test | Outils bureautiques  Ordinateur | Ordinateur |
| Spécification des Tests | Outils bureautiques  Spécifications fonctionnelles détaillées | Dernière version des documents |

Tableau 4‑1 : Ressources matérielles

# Stratégie de test

## Stratégie de validation

La stratégie de validation s’effectue en trois étapes :

* Tests unitaires ;
* Tests d’intégration ;
* Tests de validation fonctionnelle.

Les tests unitaires sont pratiqués de manière systématique au cours du développement.

Les tests de validation fonctionnelle sont effectués en fin de projet.

### Principe

L’objectif des tests de validation est d’assurer la conformité fonctionnelle de chaque module et de ses interfaces par rapport aux spécifications fonctionnelles et techniques. Il s’agit de parcourir l’ensemble des chemins du module en utilisant des jeux de valeurs valides et invalides. Pour cela, on applique des séries de scénarios correspondant aux modèles des traitements.

### Organisation de la stratégie

La stratégie doit également recouvrir toutes les spécifications. C’est pourquoi chaque fonctionnalité possèdera sa propre fiche de tests. Dans le cas où une fonctionnalité ne soit pas vérifiable sans une autre fonctionnalité, ces deux fonctionnalités seront testées ensemble dans la même fiche de tests.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | UC\_01 | UC\_02 | UC\_03 | UC\_04 | UC\_05 | UC\_06 | UC\_07 | UC\_08 | UC\_09 |
| N°1 | **X** |  |  |  |  |  |  |  |  |
| N°2 |  | **X** |  |  |  |  |  |  |  |
| N°3 |  |  | **X** |  |  |  |  |  |  |
| N°4 |  |  |  | **X** |  |  |  |  |  |
| N°5 |  |  |  |  | **X** |  |  |  |  |
| N°6 |  |  |  |  |  | **X** |  |  |  |
| N°7 |  |  |  |  |  |  | **X** |  |  |
| N°8 |  |  |  |  |  |  |  | **X** |  |
| N°9 |  |  |  |  |  |  |  |  | **X** |

Tableau 5‑1 : Organisation de la stratégie

#### Cas de tests nominaux :

##### Fiche de test n°1 : Séance allonge et assise simple de 5 minutes

xx

##### Fiche de test n°2 : Séance allongé et assise simple de 10 minutes

xx

##### Fiche de test n°2 : Séance allongé de 10 minutes + Séance assis de 10 minutes le même jour

##### Fiche de test n°2 : Séance allongé de 10 minutes + Séance assis de 10 minutes à un jour d’intervalle

##### Fiche de test n°3 : Deux séances assises de 5 minutes chacune le même jour

Xx

##### Fiche de test n°3 : Deux séances allongés de 5 minutes chacune le même jour

##### Fiche de test n°4 : Séance allongé de 5 minutes + Séance assis de 5 minutes le même jour

xx

##### Fiche de test n°5 : 2 Séances allongé de 5 minutes + 2 Séances assis de 5 minutes le même jour

xx

##### Fiche de test n°6

xx

##### Fiche de test n°7

xx

##### Fiche de test n°8

xx

##### Fiche de test n°9

Xx

#### Cas de tests de robustesse :

##### Fiche de test n°10 : Données manquantes

##### Fiche de test n°10 : Données invalides

#### Cas de tests aux limites :

##### Fiche de test n°10 : Deux jours de pratique consécutive avec une vie utilisée

##### Fiche de test n°10 : Perte de toutes les vies et réinitialisation de la série

##### Fiche de test n°10 : Récupération d’une vie après 5 jour consécutive de pratique

##### Fiche de test n°10 : Journée sans pratique complète

##### Fiche de test n°10 : Cas combinés où les vies sont gagnées et perdues au fil du temps.

##### Fiche de test n°10 : Perte de toutes les vies et réinitialisation de la série

##### Fiche de test n°10 : Deux vies perdues

##### Fiche de test n°10 : Journée sans pratique complète

### Objectif de test

Les objectifs de test associés à la validation sont identifiés comme suit :

**Domaine de l’application :**

Les tests couvrent les objectifs fonctionnels : vérification de la conformité de traitement des fonctions. Ce domaine fonctionnel ne couvre que les fonctions critiques (voir chapitre suivant).

#### Validation des procédures d’installation et d’exploitation

Concernant la validation des procédures d’installation et d’exploitation, aucune recette ne sera effectuée. A la livraison des lots correspondants, l’exécution des procédures permet de valider leur contenu.

### Niveaux de criticité

Trois niveaux de criticité sont définis : Critique, Courant et Annexe. Dans le cadre des tests de validation, l’affectation des niveaux de criticité aux fonctions est choisie selon la fréquence d’utilisation des fonctionnalités et leur impact sur la base de données.

* **Critique :** Ce niveau correspond aux fonctionnalités majeures de l’application.
* **Courant :** Ce niveau correspond aux fonctionnalités les plus utilisées en interrogation ou visualisation.
* **Annexe :** Ce niveau correspond aux fonctionnalités n’ayant pas d’impact sur le traitement des données et aux fonctionnalités peu utilisées.

### Critères d’arrêt des tests

#### Normal

Le critère d’arrêt normal des tests (au niveau de chaque campagne) est défini par le respect des conditions suivantes :

* Tous les cas de tests ont été déroulés ;
* Tous les résultats obtenus sont analysés et enregistrés ;
* Toutes les fiches d’anomalies sont soldées.

#### Anormal

Le critère d’arrêt anormal d’une campagne intervient dans les cas suivants :

* Ressources nécessaires à la campagne insuffisantes ;
* Déroulement non conforme ;
* Nombre d’anomalies détectées dans l’exécution trop important.

Le premier cas est contrôlé lors de la préparation de l’exécution des tests. Les deux autres cas sont traités directement dans les scénarios de test par le contrôle et l’enregistrement des fichiers.

## Circuit des modifications

Le circuit des modifications peut intervenir lors de chaque phase d’activité de test :

* Spécification
* Conception
* Exécution
* Analyse

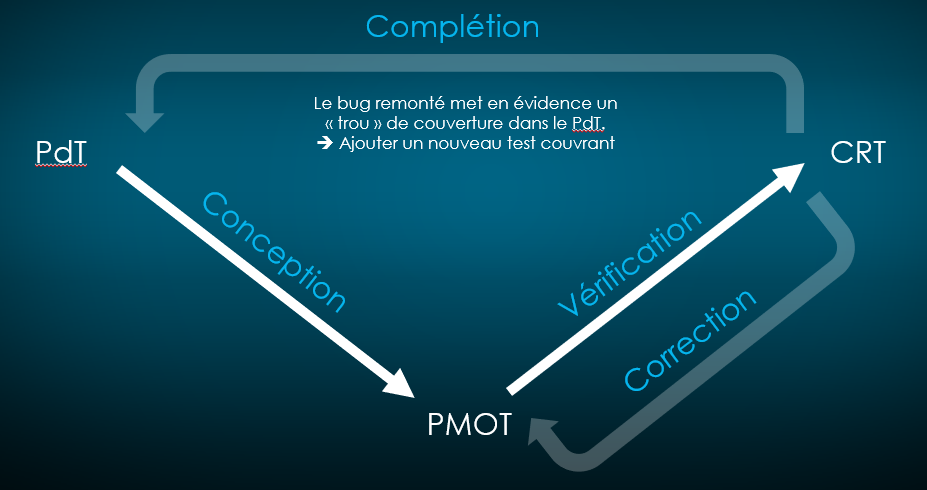


Figure 5‑2 : Convergence de la Qualité par Amélioration Continue de R.Bourdillon

Il permet la prise en compte des anomalies rencontrées dans la phase de test impliquée :

* Anomalie de spécification
* Anomalie de conception
* Anomalie d’exécution

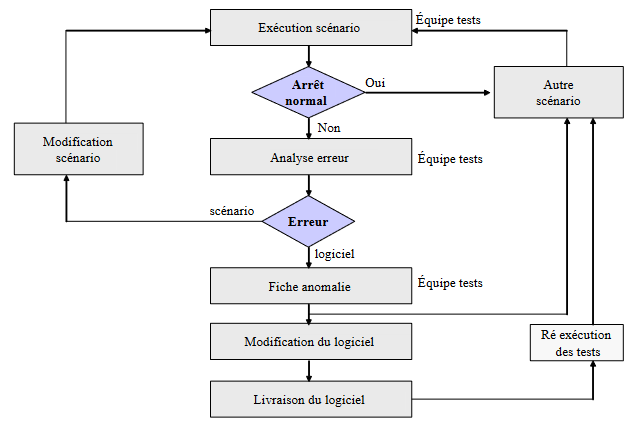


Figure 5‑3 : Cycle de prise en compte

Le suivi de ces anomalies sera traité par le chef de projet ainsi que par le responsable qualité.

# Description des tests

## Le cycle de développement

Le cycle de développement logiciel en vigueur est le suivant :

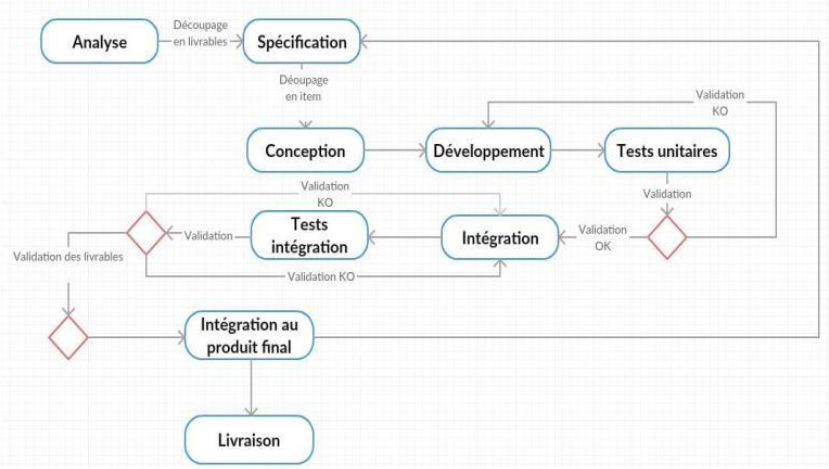


Figure 6‑1 : Cycle de développement logiciel en vigueur

## Validation des projets

### Dossier de tests unitaires

Les tests unitaires sont effectués par le programmeur lors du développement. Les anomalies non corrigées feront l’objet d’un recensement dans un tableau de suivi.

### Dossier de tests d’intégration

Le dossier de tests d’intégration est initialisé dans la phase de conception générale et finalisé avant la phase de tests d’intégration. Il contient l’ensemble des tests garantissant l’intégration des modules entre eux puis dans l’environnement définitif.

### Dossier de tests de validation

Le dossier de tests de validation est initialisé dans les phases de spécification et finalisé avant la phase de validation. On effectuera en priorité les tests de stabilité pour vérifier les performances globales du logiciel.