

CAHIER DE RECETTES

**Projet Maths-Infos 2017-2018**

Version: 1.3

Auteur: Romain JACQUIEZ, Antoine BERENGUER et Théo HIPAULT

ISEN Toulon - Yncrea  
Maison du Numérique et de l'Innovation  
Place Georges Pompidou

83000 Toulon

**Description du document**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Type** | | **Version** | | **Confidentialité** | | |
| Cahier de recette | | 1.3 | | Usage externe | | |
|  | **Nom** | | **Fonction** | | **Date** | **Visa** |
| **Rédacteur** | Romain JACQUIEZ | | Rédaction | | 19/12/17  au  22/12/17 |  |
| **Vérificateur** | Romain JACQUIEZ, Théo HIPAULT | | Corrections | | 22/12/17 |  |
| **Approbateur** | M.THELOT, M.FRESSEL | | Validation | | 22/12/17 |  |
| **Destinataire** | | **Fonction** | | **Organisme** | | |
| Client | | Lecture | | ISEN | | |

**Révisions du document**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Rédacteur** | **Modifications** |
| 1.0 | 18/07/2016 | FMC | Mise en forme |
| 1.1 | 18/12/2017 | TH - RJ - AB | Liste des exigences fonctionnelles et de design |
| 1.2 | 19/12/2017 | TH - AB | Rédaction des fiches de tests associés |
| 1.3 | 21/12/2017 | TH | Correction des exigences fonctionnelles |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Sommaire**

[**L’INTRODUCTION**](#_gjdgxs)5

[**LES EXIGENCES**](#_9goux5qul3uu)7

[**DESCRIPTION DES TESTS FONCTIONNELS**](#_30j0zll)8

**Index des illustrations**

**Index des tables**

**REFERENCES**

|  |  |  |
| --- | --- | --- |
| **Référence** | **Description** | **Nom** |

|  |  |  |
| --- | --- | --- |
| [1] |  |  |

|  |  |  |
| --- | --- | --- |
| [2] |  |  |

**DEFINITIONS**

Sans objet

**ABBREVIATIONS**

ISEN : Institut Supérieur de l'Electronique et du Numérique

# L’INTRODUCTION

La **recette d'un logiciel** consiste à vérifier si le produit obtenu à la fin du processus de développement est conforme aux attentes. Le cahier de recette décrit précisément les tests à réaliser pour réaliser la recette du logiciel.

Les concepts déjà abordés dans le cahier des charges se retrouvent liés aux concepts utilisés dans le cahier de recette. Ainsi nous pouvons dire:

* qu'une **exigence** dont il faut s'assurer du bon fonctionnement sera associée à
* un **cas de de test** (ensemble des tests à réaliser afin de vérifier tout ou une partie d’une exigence), qui sera décrit dans
* une **fiche de tests** (description technique de la manière le cas sera vérifié, elle comporte le mode opératoire des opérations techniques à faire pour obtenir les résultats à vérifier), qui fera partie
* d'un **scénario** (ensemble de fiches de tests ordonnées avec une logique opératoire) associé à
* une **campagne de tests** (ensemble de scénarios exécutés sur un même environnement et par une même équipe) qui fera partie
* d'une **recette** (ensemble de campagnes de tests)

Il existe différents types de test , parmi lesquels :

* les **tests unitaires techniques**, qui sont réalisés par le programmeur, la plupart du temps à l'aide d'un cadriciel dédié comme CUnit ou JUnit, et qui permettent de valider un module fonction par fonction, afin de s'assurer qu'il est conforme aux spécifications fonctionnelles. Les test unitaires permettent notamment de vérifier que l'évolution du programme n'entraîne pas de régression.

Ces tests s'appuient sur la connaissance précise des composants internes : boîte blanche.

* les **tests d’intégration**, qui permettent de valider le bon fonctionnement de toutes les parties du programme développées indépendamment.
* les **tests de performance**, qui portent sur le temps d'exécution et la consommation mémoire.
* les **tests de non régression** qui sont à effectuer lors de l'évolution du logiciel, afin de vérifier que ses fonctionnalités antérieures soient conservées à l'identique.
* les **tests de validation**, qui doivent vérifier que les exigences fonctionnelles et techniques ont été respectées.

Ces tests ignorent la connaissance des composants internes : boîte noire.

La recette peut être:

* "**usine**", c'est à dire réalisée par le fournisseur du logiciel.

Recette usine = tests unitaires + d'intégration + de validation + de non régression.

* ou bien "**utilisateur**", si elle est réalisée par le client.

La **recette fonctionnelle** validera les les spécifications exprimées dans le  
 cahier des charges. La **recette technique** validera les caractéristiques techniques  
 du produit livré, afin de s'assurer qu'il répond en terme de performances et  
 d'exploitation à l'environnement dans lequel il sera utilisé.

Recette utilisateur = tests de validation + tests de performance.

|  |
| --- |
| LES EXIGENCES |

Ex\_xxxx\_O : Exigence obligatoire

Ex\_xxxx\_F : Exigence facultative

**Exigences fonctionnelles :**

EF\_0001\_O : Le programme est écrit et compilé en langage Java

EF\_0002\_O : Le programme effectue le calcul de la FFT d’un signal réel

EF\_0003\_O : Le programme effectue le calcul de la FFT d’un signal complexe

EF\_0004\_O : Le programme effectue le calcul de la FFT inverse d’un signal complexe

EF\_0005\_F : Une classe permettant de gérer des nombres complexes est implémentée

EF\_0006\_O : Les signaux traités sont de taille N, une puissance de 2 choisie par l’utilisateur

EF\_0007\_O : Des signaux types permettant de tester facilement le programme sont proposés

EF\_0008\_F : Un signal peut être traité à partir d’un fichier .csv

EF\_0009\_F : Les résultats des calculs sont enregistrés dans un fichier .csv

**Exigences de design :**

ED\_0001\_O : Le programme s’affiche de façon conviviale dans une console

ED\_0002\_O : Les choix et entrées de l’utilisateur sont recueillis au moyen d’un menu

ED\_0003\_F : Le signal de départ est affiché sous forme d’un graphique dans une fenêtre

ED\_0004\_F : Le signal après FFT est affiché sous forme d’un graphique dans une fenêtre

ED\_0005\_F : Le signal après FFT inverse est affiché sous forme d’un graphique dans une fenêtre

# DESCRIPTION DES TESTS FONCTIONNELS

L'identifiant de test sera de la forme TEST\_Ex\_xxxx\_x afin de permettre de se référer facilement à l'exigence concernée.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fiche de Test Fonctionnel :** [TEST\_EF\_0001\_O ] | | | | | |
| **Objectif :** | | Vérifier que le programme est écrit en langage Java | | | |
|  | |  | | | |
| **Exigence à tester :** | | EF\_0001\_O | | | |
|  | |  | | | |
| **Pré-requis :** | | Code source du programme | | | |
|  | |  | | | |
| **Initialisation :** | |  | | | |
|  | |  | | | |
| **Etapes du test :** | | | | | |
| **Id** | **Démarche** | | **Données** | **Comportement attendu** | **OK?** |
| **1** | Reconnaître le langage utilisé | |  | Le langage utilisé est Java | Oui |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fiche de Test Fonctionnel :** [TEST\_EF\_0002\_O] | | | | | |
| **Objectif :** | | Vérifier que le programme peut calculer la FFT d’un signal réel | | | |
|  | |  | | | |
| **Exigence à tester :** | | EF\_0002\_O | | | |
|  | |  | | | |
| **Pré-requis :** | |  | | | |
|  | |  | | | |
| **Initialisation :** | |  | | | |
|  | |  | | | |
| **Etapes du test :** | | | | | |
| **Id** | **Démarche** | | **Données** | **Comportement attendu** | **OK?** |
| **1** | Lancer la fonction de test JUnit5 FFTrTest() dans la classe FourierTest | |  | Le test est réussi | Oui |
| **2** | Démarrer le programme | | Retour console | Un menu s’affiche en console | Oui |
| **3** | Rentrer “0” dans la console pour choisir l’option “0 - Signal incrémenté” | | Retour console | Une phrase s’affiche pour demander la taille du signal désiré | Oui |
| **4** | Rentrer “16” dans la console, ou une autre taille qui est une puissance de 2.  Constater les résultats | | Retour console et graphique | Les résultats s’affichent en console et trois graphes s’affichent dans des fenêtres. Les données correspondent à la FFT d’un signal incrémenté | Oui |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fiche de Test Fonctionnel :** [TEST\_EF\_0003\_O ] | | | | | |
| **Objectif :** | | Vérifier que le programme peut calculer la FFT d’un signal complexe | | | |
|  | |  | | | |
| **Exigence à tester :** | | EF\_0003\_O | | | |
|  | |  | | | |
| **Pré-requis :** | | Code source du programme | | | |
|  | |  | | | |
| **Initialisation :** | |  | | | |
|  | |  | | | |
| **Etapes du test :** | | | | | |
| **Id** | **Démarche** | | **Données** | **Comportement attendu** | **OK?** |
| **1** | Lancer le test Junit FFTcTest() dans la classe FourierTest | |  | Le test est réussi | Oui |
| **2** | Démarrer le programme après avoir changé l’appel de FFTr à FFTc | | Retour console | Un menu s’affiche en console | Oui |
| **3** | Rentrer “4” dans la console pour choisir l’option “4 - Impulsion de Dirac” | | Retour console | Une phrase s’affiche pour demander la taille du signal désiré | Oui |
| **4** | Rentrer “16” dans la console, ou une autre taille qui est une puissance de 2.  Constater les résultats | | Retour console et graphique | Les résultats s’affichent en console et trois graphes s’affichent dans des fenêtres. Les données correspondent à la FFT d’un signal complexe | Oui |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fiche de Test Fonctionnel :** [TEST\_EF\_0004\_O ] | | | | | |
| **Objectif :** | | Vérifier que le programme effectue le calcul de la FFT inverse d’un signal complexe | | | |
|  | |  | | | |
| **Exigence à tester :** | | EF\_0004\_O | | | |
|  | |  | | | |
| **Pré-requis :** | | Code source du programme | | | |
|  | |  | | | |
| **Initialisation :** | | Démarrer le programme | | | |
|  | |  | | | |
| **Etapes du test :** | | | | | |
| **Id** | **Démarche** | | **Données** | **Comportement attendu** | **OK?** |
| **1** | Lancer le test Junit iFFTTest() dans la classe FourierTest | |  | Le test est réussi | Oui |
| **2** | Démarrer le programme | | Retour console | Un menu s’affiche en console | Oui |
| **3** | Rentrer “4” dans la console pour choisir l’option “4 - Impulsion de Dirac” | | Retour console | Une phrase s’affiche pour demander la taille du signal désiré | Oui |
| **4** | Rentrer “16” dans la console, ou une autre taille qui est une puissance de 2.  Constater les résultats | | Retour console et graphique | Les résultats s’affichent en console et trois graphes s’affichent dans des fenêtres. Les données correspondent à la iFFT d’un signal incrémenté | Oui |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fiche de Test Fonctionnel :** [TEST\_EF\_0005\_F ] | | | | | |
| **Objectif :** | | Vérifier qu’une classe permettant de gérer des nombres complexes est implémentée | | | |
|  | |  | | | |
| **Exigence à tester :** | | EF\_0005\_F | | | |
|  | |  | | | |
| **Pré-requis :** | |  | | | |
|  | |  | | | |
| **Initialisation :** | |  | | | |
|  | |  | | | |
| **Etapes du test :** | | | | | |
| **Id** | **Démarche** | | **Données** | **Comportement attendu** | **OK?** |
| **1** | Lancer le test JUnit5 NombreComplexeTest | |  | Les 5 tests sont réussis | Oui |
| **2** | Lancer le test JUnit5 OperationComplexeTest | |  | Les 6 tests sont réussis | Oui |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fiche de Test Fonctionnel :** [TEST\_EF\_0006\_O ] | | | | | |
| **Objectif :** | | Vérifier que les signaux traités sont de taille N, une puissance de 2 choisie par l’utilisateur | | | |
|  | |  | | | |
| **Exigence à tester :** | | EF\_0006\_O | | | |
|  | |  | | | |
| **Pré-requis :** | | Code source du programme | | | |
|  | |  | | | |
| **Initialisation :** | | Démarrer le programme | | | |
|  | |  | | | |
| **Etapes du test :** | | | | | |
| **Id** | **Démarche** | | **Données** | **Comportement attendu** | **OK?** |
| **1** | Démarrer le programme | | Retour console | Un menu s’affiche en console | Oui |
| **2** | Choisir un signal parmi les choix proposés | | Retour console | Une phrase s’affiche pour demander la taille du signal désiré | Oui |
| **3** | Entrer une taille qui n’est pas une puissance de 2 (exemple: -12) | | Retour console | Le programme signale la valeur erronée à l’utilisateur et le propose de rentrer une autre valeur | Oui |
| **4** | Entrer une taille qui est une puissance de 2 (exemple: 16) | | Retour console | Le programme exécute les fonctions de Fourier avec les paramètres donnés | Oui |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fiche de Test Fonctionnel :** [TEST\_EF\_0007\_O ] | | | | | |
| **Objectif :** | | Vérifier que des signaux types sont proposés par le programme | | | |
|  | |  | | | |
| **Exigence à tester :** | | EF\_0007\_O | | | |
|  | |  | | | |
| **Pré-requis :** | |  | | | |
|  | |  | | | |
| **Initialisation :** | | Démarrer le programme | | | |
|  | |  | | | |
| **Etapes du test :** | | | | | |
| **Id** | **Démarche** | | **Données** | **Comportement attendu** | **OK?** |
| **1** | Constater la présence de 5 signaux types | | Retour console | Un menu proposant plusieurs signaux s’affiche | Oui |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fiche de Test Fonctionnel :** [TEST\_EF\_0008\_F] | | | | | |
| **Objectif :** | | Vérifier q’un signal peut être traité à partir d’un fichier .csv | | | |
|  | |  | | | |
| **Exigence à tester :** | | EF\_0008\_F | | | |
|  | |  | | | |
| **Pré-requis :** | | Code source du programme | | | |
|  | |  | | | |
| **Initialisation :** | | Démarrer le programme | | | |
|  | |  | | | |
| **Etapes du test :** | | | | | |
| **Id** | **Démarche** | | **Données** | **Comportement attendu** | **OK?** |
| **1** | Démarrer le programme | | Retour console | Un menu s’affiche en console | Oui |
| **2** | Rentrer “5” dans la console pour choisir l’option “5 - Import de CSV” | | Retour console | Une phrase s’affiche pour demander la taille du signal désiré | Oui |
| **3** | Rentrer “16” dans la console, ou une autre taille qui est une puissance de 2.  Constater les résultats | | Retour console et graphique | Les résultats s’affichent en console et trois graphes s’affichent dans des fenêtres. | Oui |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fiche de Test Fonctionnel :** [TEST\_EF\_0009\_F ] | | | | | |
| **Objectif :** | | Vérifier que les résultats des calculs sont enregistrés dans un fichier .csv | | | |
|  | |  | | | |
| **Exigence à tester :** | | EF\_0009\_F | | | |
|  | |  | | | |
| **Pré-requis :** | | Créer un fichier .csv conforme (voir Introduction) | | | |
|  | |  | | | |
| **Initialisation :** | | Démarrer le programme | | | |
|  | |  | | | |
| **Etapes du test :** | | | | | |
| **Id** | **Démarche** | | **Données** | **Comportement attendu** | **OK?** |
| **1** | Rentrer “5” en console pour sélectionne l’option “5 - Import de CSV” | | Retour console | Une phrase s’affiche pour demander la taille du signal désiré | Oui |
| **2** | Rentrer la taille du signal du fichier .csv, toujours une puissance de 2 | | Retour console et fenêtre de sélection de fichier | Une fenêtre de sélection de fichier s’affiche | Oui |
| **3** | Naviguer jusqu’au fichier .csv et valider | | Retour console et fenêtres graphiques | Les résultats des calculs sont affichés dans la console et trois graphes sont affichés dans des fenêtres | Oui |

|  |  |  |
| --- | --- | --- |
| **Exigences de design :** | | |
| **ID** | **Description** | **OK?** |
| ED\_0001\_O | Le programme s’affiche de façon conviviale dans une console | Oui |
| ED\_0002\_O | Les choix et entrées de l’utilisateur sont recueillis au moyen d’un menu | Oui |
| ED\_0003\_F | Le signal de départ est affiché sous forme d’un graphique dans une fenêtre | Oui |
| ED\_0004\_F | Le signal après FFT est affiché sous forme d’un graphique dans une fenêtre | Oui |
| ED\_0005\_F | Le signal après FFT inverse est affiché sous forme d’un graphique dans une fenêtre | Oui |