CAHIER DE RECETTE

Version :	*1.0*
Date :	*28/05/14*
Rédigé par :	**Tony Coriolle**
Relu par :	*Julien Szlamowicz, Delphine Meyrieux*
Signature :	

MISES A JOUR

Version	Date	Modifications réalisées
0.1	24/11/13	Création
0.2	16/01/14	Modifications suite à recommandations
0.3	10/03/14	Ajout des tests issus du redécoupage de taches de la STB + validation des tests
0.4	10/04/14	Validations des tests pour le sprint 2 + ajout section livrable
1.0	28/05/14	Version finale

Table des matières

1. Introduction :

Ce document nous permettra de définir les moyens et les procédés (tests) mis en œuvre pour assurer la validation du produit. L'objectif de la recette est de vérifier que le logiciel est conforme aux attentes exprimées dans les spécifications techniques du besoin.

Le logiciel pourra permettre à l'utilisateur de :

- *faire le choix entre deux modes de fonctionnement, à savoir un mode de calcul sérialisé (SAGE) ou parallélisé (CUDA)*
- *s'informer sur l'algorithme qu'il veut utiliser*
- *choisir l'algorithme à utiliser*
- *définir le nombre qu'il souhaite factoriser*
- *définir la base du nombre entré (hex, bin, dec)*
- *afficher un rapport d'exécution (XML)*
- *comparer deux rapports d'exécution*

Les objets à tester seront :

- *Validité de l'Algorithme de Dixon*
- *Vérifier le comportement de l'IHM*
- *Génération XML*
- *Comparaison de rapports*
- *Validité des entrées et sorties de toutes les fonctions*

2. Documents applicables et de référence

Les documents de référence seront :

- *La spécification technique du besoin (v0.3)*
- *Tutoriel de test unitaire avec CxxTest : http://web-cat.cs.vt.edu/eclipse/cxxtest/*
- *Guide utilisateur de CxxTest : http://cxxtest.com/guide.html*
- *Aide python pour unittest : http://docs.python.org/2/library/unittest.html*

3. Terminologie et sigles utilisés

Voir document Terminologie

4. Environnement de test

- *Tests unitaires :*

Les tests seront effectués sur nos machines personnelles, aucune contrainte de disponibilité, accessibilité, etc. n'est à envisager.

- *Tests d'intégration :*

Réalisation soit depuis les machines de l'université lors des réunions de l'équipe soit depuis nos machines si les disponibilités de chacun ne nous permettent pas de se réunir au moment de la livraison des composants.

- *Tests fonctionnels :*

La réalisation de ces tests sera faite sur la machine mise à notre disposition à l'université afin de réaliser les tests fonctionnels de la partie CUDA. Les tests de la partie Sage et sur l'IHM seront eux réalisés depuis nos machines. La salle de projet ne nous étant pas réservé, la réalisation des tests pour CUDA devra être faite en fonction de la disponibilité de la salle.

L'ensemble des machines utilisables, ainsi que leurs configurations sont définies dans le Dossier Architecture Logiciel.

Le langage SAGE embarque un module de tests unitaires qui sera utilisé. Ce module se nomme « instance_tester » et est disponible dans la librairie « sage.misc.sage_unittest ».

La deuxième partie sera développée en C++ ce qui impliquera de faire des tests unitaires avec le Framework « CxxTest ».

Le jeu de données sera composé de grands entiers qui devront être mis en place par nos soins. *« TeamCity », un outil d'intégration continue sur internet, nous permettra de déclarer des scénarios de tests, ainsi que de rassembler l'ensemble des tests unitaires des membres de l'équipe. Il nous permettra aussi de savoir quand une version est stable.*

5. Responsabilités

Chaque développeur aura la responsabilité de réaliser les tests unitaires de chaque fonctionnalité implémentée, puis le responsable qualité vérifiera les tests existants et ajoutera si nécessaire d'autres tests unitaires sur les fonctionnalités livrées. Chacun sera libre de rajouter ses données de test dans la base de données de test.

Le responsable qualité devra automatiser au maximum les tests d'intégration qui seront effectués via TeamCity, cela nous permettra d'avoir une intégration continue, d'identifier rapidement les parties de code défaillantes et de valider ou non les versions à mettre en recette.

6. Stratégie de tests

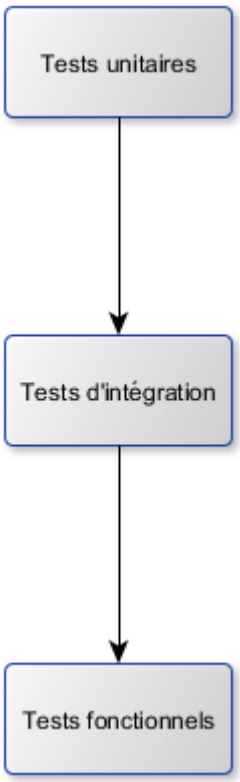

Figure 1 - campagne de test

Lorsqu'une fonctionnalité sera dite terminée la campagne de tests pourra débuter.

La campagne de test se déroulera de la façon suivante : Chaque fonctionnalité devra subir des tests unitaires ce qui lui permettra de pouvoir accéder à la deuxième étape des tests : l'intégration. Une fois les tests d'intégration réussis, le composant sera considéré comme valide et intégré au projet puis nous pourrons recommencer ce schéma de tests sur toute autre fonctionnalité non encore validé.

Lorsque l'application sera complète et tous les tests d'intégrations passés, des tests fonctionnels seront mis en place pour vérifier le bon fonctionnement de l'application.

Un test commencera par l'initialisation ou la récupération des données depuis le jeu de données afin de paramétrer si nécessaire la fonction ou le composant à tester. La fonction sera ensuite exécuté, puis en fonction du code de retour le test sera passé ou non.

Ceci nous permettra de dire que le test est validé, ainsi nous pourrons passer au test suivant

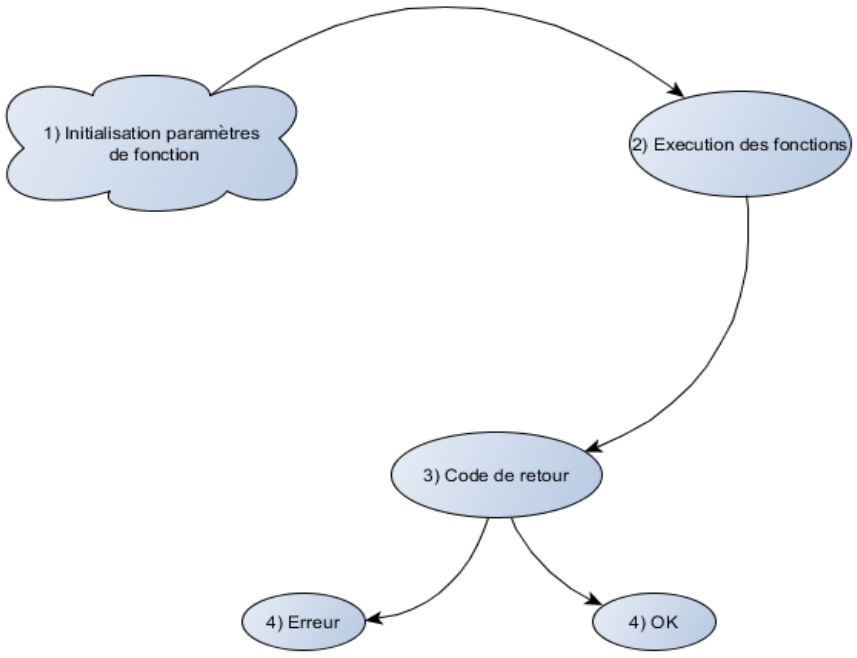

Figure 2 - schéma d'exécution d'un test

7. Gestion des anomalies

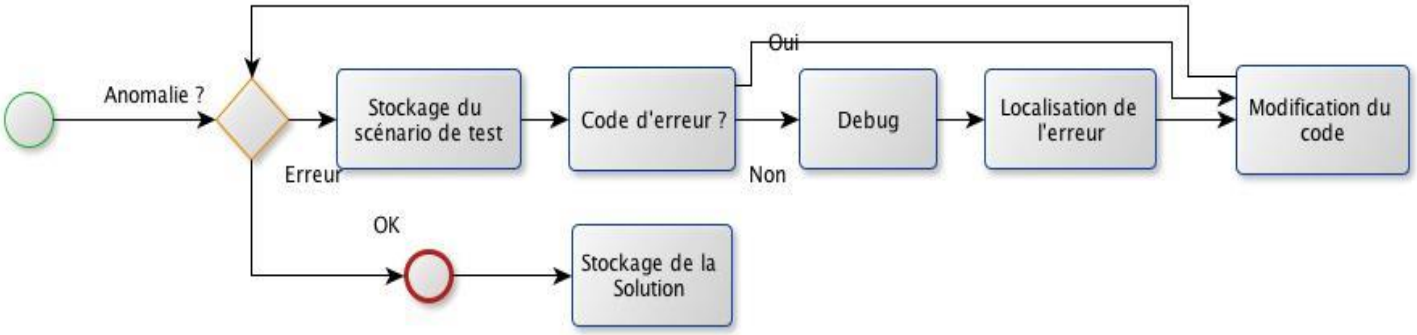

Figure 3 - Gestion des erreurs

Lors de la phase de développement chaque membre de l'équipe aura la responsabilité d'ajouter d'éléments de debug qui permettront de pouvoir réaliser les tests comme ci-dessus (date d'anomalie, code d'erreur, commentaire).

Lorsqu'une anomalie est détectée lors d'un test unitaire ou test d'intégration, le scénario de test devra être conservé afin de pouvoir tester ce même scénario une fois la solution apportée.

Puis si un code d'erreur est retourné ce dernier permettra de localiser l'erreur rapidement d'appliquer un patch, sinon la phase de debug débutera jusqu'à isolation de l'erreur et modification du code.

Une fois le patch appliqué le test devra être relancé, si aucune erreur n'est détectée la solution sera stockée ainsi qu'une trace de l'anomalie, sinon le schéma se répètera.

Ceci nous permettra par la suite d'aller vérifier cette base si une erreur similaire réapparait.

8. Procédures de test

Dans ce paragraphe nous spécifierons les tests qui devront être mis en place afin de valider les composants faisant partie de la spécification du besoin.

Chaque composant se verra attribuer un numéro qui nous permettra de les différencier. Dans l'en-tête des procédures l'objet et l'objectif du test seront spécifiés tout comme les préconditions qui s'appliquerons sur cette procédure.

Test fonctionnel : Test de navigation	*Version : **1.0***				
Objectif du test : Déplacement dans le menu					
Objet testé : Interface Graphique					
***Procédure n° [Test** UI – 1]*					
Précondition : aucune					
N°	***Actions***	***Résultats attendus***	***Conditions d'arrêt valide***	***Arrêt suite à erreur***	***OK / NOK ?***
1	Cliquer sur les menus afin de se déplacer dans les sous menus	Obtenir le menu désiré lors du choix.			***OK***
2	Survoler la liste des algorithmes	Obtenir les informations désirées sur les algorithmes disponibles			***NOK***
3	Saisir un nombre puis l'enregistrer	Nombre stocké en mémoire avec conversion si nécessaire			***OK***
Validé ? Signature					

Test fonctionnel : Rapport	*Version : **1.0***				
Objectif du test : utilisation de la bonne heuristique					
Objet testé : Interface Graphique					
***Procédure n° [Test** UI – 2]*					
Précondition : heuristique sélectionnée, nombre enregistré					
N°	***Actions***	***Résultats attendus***	***Conditions d'arrêt valide***	***Arrêt suite à erreur***	***OK / NOK ?***
1	Lancer l'exécution de l'heuristique	Affichage de la page de surveillance de l'exécution	*Valeur affichées et correctes*	*Valeurs inconnues ou incorrectes*	***NOK***
2	Vérifier dans le rapport d'exécution la valeur des champs heuristique	Champs heuristique correspondant à la valeur de départ			***NOK***
Validé ? Signature					

Test fonctionnel : Test de sauvegarde	*Version : **1.0***				
Objectif du test : sauvegarde du nombre à factoriser					
Objet testé : Interface Graphique					
***Procédure n° [Test** UI – 3]*					
Précondition : aucune					
N°	***Actions***	***Résultats attendus***	***Conditions d'arrêt valide***	***Arrêt suite à erreur***	***OK / NOK ?***
1	Entrer un nombre à factoriser à l'endroit prévu	Affichage du nombre dans la boite			***OK***
2	Enregistrer ce nombre	Nombre stocké en mémoire avec conversion si nécessaire	*Valeur correcte stocké*	*Valeur incorrecte*	***OK***
Validé ? Signature					

Test fonctionnel : Rapport	*Version : **1.0***				
Objectif du test : Affichage d'une description					
Objet testé : Interface Graphique					
***Procédure n° [Test** UI – 4]*					
Précondition : aucune					
N°	***Actions***	***Résultats attendus***	***Conditions d'arrêt valide***	***Arrêt suite à erreur***	***OK / NOK ?***
1	Survoler un algorithme	Affichage de toutes les informations concernant l'algorithme survolé			***NOK***
Validé ? Signature					

Test fonctionnel : Rapport	*Version : **1.0***				
Objectif du test : Observation d'un changement d'état					
Objet testé : Interface Graphique					
***Procédure n° [Test** UI – 5]*					
Précondition : nombre enregistré et exécution en cours					
N°	***Actions***	***Résultats attendus***	***Conditions d'arrêt valide***	***Arrêt suite à erreur***	***OK / NOK ?***
1	Attente	Affichage de toutes les valeurs des variables issus de l'exécution			***NOK***
Validé ? Signature					

Test fonctionnel : Rapport	*Version : **1.0***				
Objectif du test : Valeur du rapport correspondant à l'affichage					
Objet testé : Interface Graphique					
***Procédure n° [Test** UI – 6]*					
Précondition : traitement d'un nombre lancé et exécution terminé					
N°	***Actions***	***Résultats attendus***	***Conditions d'arrêt valide***	***Arrêt suite à erreur***	***OK / NOK ?***
1	Demande de l'affichage d'un rapport	Affichage de toutes les valeurs des variables issus de l'exécution	*Valeur affichées et correctes*	*Valeurs inconnues ou incorrectes*	***OK***
Validé ? Signature					

Objet testé : ***SAGE***	*Version :* ***1.0***
Objectif de test : Bon retour des valeurs de l'algorithme de Dixon	
Procédure n° [Test SG - 1]	
Préconditions : • ***Spécifier un N entier très grand*** • ***Une borne B*** • ***La base de référence***	

N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	***Arrêt suite à erreur***	***OK / NOK ?***
1	Déclencher l'initialisation de l'algorithme	Récupération du nombre et de la borne passés en entrée, conversion si nécessaire du nombre			***OK***
2	Exécution de l'algorithme	Produit de facteurs premiers égal à l'entier de départ (test logiciel : fonction is_prime() en SAGE)	*Facteurs premiers*	*Facteurs premiers* *Erreur système*	***OK***

Validé ? Signature	

Objet testé : ***SAGE***	*Version :* ***1.0***
Objectif de test : Vérifier le bon fonctionnement de la passerelle	
Procédure n° [Test *SG – 2]*	
Précondition :	

N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	***Arrêt suite à erreur***	***OK / NOK ?***
1	Lancement de l'exécution depuis l'IHM	Affichage de la page de surveillance de l'exécution			***OK***

Validé ? Signature	

Objet testé : ***SAGE***	*Version :* ***1.0***
Objectif de test : Bon retour des valeurs des heuristiques	
***Procédure n° [Test** SG – 3]*	
Précondition :	

N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	***Arrêt suite à erreur***	***OK / NOK ?***
1	Déclencher l'initialisation de l'algorithme	Récupération du nombre et de la borne passés en entrée			***OK***
2	Exécution de l'algorithme	Produit de facteurs premiers égal à l'entier de départ (test logiciel : fonction is_prime() en SAGE)			***OK***

Validé ? Signature	

Objet testé : ***CUDA***			*Version :* ***1.0***		
Objectif de test : Valeur du pgcd					
***Procédure n° [Test** CD – 1]*					
Précondition : nombre passé en entré					
N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	*Arrêt suite à erreur*	***OK / NOK ?***
1	Déclencher l'initialisation de l'algorithme	Récupération d'une seule valeur			***OK***
2		Valeurs correctes dans la liste			***OK***
Validé ? Signature					

Objet testé : ***CUDA***			*Version :* ***1.0***		
Objectif de test : liste des premiers					
***Procédure n° [Test** CD – 2]*					
Précondition : borne passé en entré					
N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	*Arrêt suite à erreur*	***OK / NOK ?***
1	Déclencher l'initialisation de l'algorithme	Récupération d'une liste de premiers			***OK***
2		Valeurs correctes dans la liste			***OK***
Validé ? Signature					

Objet testé : ***CUDA***	*Version :* ***1.0***
Objectif de test : Ensemble R correctement rempli	
***Procédure n° [Test** CD – 3]*	
Précondition : Ensemble r vide, n le nombre à factoriser, la borne, ensemble div (vide ou non) , la taille de div	

N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	*Arrêt suite à erreur*	***OK / NOK ?***
1	Exécution de l'algorithme	R est rempli			***OK***

Validé ? Signature	

Objet testé : ***CUDA***	*Version :* ***1.0***
Objectif de test : valeur du produit de l'ensemble P correct	
***Procédure n° [Test** CD – 4]*	
Précondition : CD-3 effectuée	

N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	*Arrêt suite à erreur*	***OK / NOK ?***
1	Exécution de l'algorithme	Génération d'une matrice des vecteurs vi tels que yi = au produit des nombres premiers p exposant vi			***OK***

Validé ? Signature	

Objet testé : ***CUDA***	*Version :* ***1.0***
Objectif de test : Bon retour des valeurs de l'algorithme de Dixon	
Procédure n° [Test CD - 5]	
Préconditions : • ***Spécifier un N entier très grand*** • ***Une borne B*** • ***La base de référence***	

N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	***Arrêt suite à erreur***	***OK / NOK ?***
1	Déclencher l'initialisation de l'algorithme	Récupération du nombre et de la borne passés en entrée			***OK***
2	Exécution de l'algorithme	Produit de facteurs premiers égal à l'entier de départ (test réalisé au sprint 3)	Facteurs premiers	Erreur système	OK

Validé ? Signature	

Objet testé : ***CUDA***	*Version :* ***1.0***				
Objectif de test : Heuristiques					
***Procédure n° [Test** CD – 6]*					
Précondition :					
N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	*Arrêt suite à erreur*	***OK / NOK ?***
1	Déclencher l'initialisation de l'algorithme	Récupération du nombre et de la borne passés en entrée			***NOK***
2	Exécution de l'algorithme	Produit de facteurs premiers égal à l'entier de départ (test réalisé au sprint 3)			***NOK***
Validé ? Signature					

Objet testé : ***Options***			*Version :* ***1.0***		
Objectif de test : Génération de XML					
***Procédure n° [Test** MC – 1]*					
Précondition : factorisation terminée					
N°	***Actions***	***Résultats attendus***	***Conditions d'arrêt valide***	***Arrêt suite à erreur***	***OK / NOK ?***
1	Demande d'un rapport d'exécution	Récupération des données concernant la factorisation : temps d'exécution, liste des facteurs avec leurs puissances, le nombre d'instructions effectuées, l'entier de départ, l'algorithme utilisé, la méthode utilisée et les caractéristiques matérielles.		*Pas de Données* *Erreur de récupération*	 ***OK***
2	Génération XML	Rapport XML contenant les données de 1.	*Fichier généré*		***OK***
Validé ? Signature					

Objet testé : ***Options***	*Version :* ***1.0***
Objectif de test : Comparaison de fichiers XML	
***Procédure n° [Test** MC – 2]*	
Précondition : • ***factorisations terminées*** • ***même nombre factorisé*** • ***rapports au format XML***	

N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	***Arrêt suite à erreur***	***OK / NOK ?***
1	Référencer les rapports à examiner	Stockage en mémoire des deux rapports		Fichiers non reconnus Format de fichier invalide Entiers différents	**OK**
2	Lancer l'examen des rapports	Ce que le logiciel doit fournir comme résultat.	Rapport généré		***OK***

Validé ? Signature	

Objet testé : ***Options***	*Version :* ***1.0***
Objectif de test : mise en pause processus	
***Procédure n° [Test** MC – 3]*	
Précondition :	

N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	***Arrêt suite à erreur***	***OK / NOK ?***
1	Mettre en pause le programme à n'importe quel moment du calcul	Arrêt des calculs			***NOK***
2	Reprise du programme	Le programme reprend à l'endroit où il s'est arrêté		Perte de l'ordre des calculs Valeurs Incorrectes	***NOK***

Validé ? Signature	

Objet testé : ***Options***	*Version :* ***1.0***
Objectif de test : Choix d'une base	
***Procédure n° [Test** MC – 5]*	
Précondition :	

N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	***Arrêt suite à erreur***	***OK / NOK ?***
1	Indiquer la base de représentation	Stockage en mémoire des deux rapports	Valeur correcte	Erreur de conversion	OK

Validé ? Signature	

9. Livrable

Les livrables contiendrons le code source ainsi qu'une documentation de chaque fonction, une documentation expliquant le fonctionnement de l'application Sage et une autre expliquera le fonctionnement de l'application Cuda. Un makefile pourra permettre la compilation du programme C/Cuda afin de faciliter la tâche au client.

Version : 1.0

	SAGE	CUDA
Code Sources	X	X
Manuel de conception	X	X
Manuel d'utilisation	X	X
Makefile	X	X