CAHIER DE RECETTE

Version :	*0.1*
Date :	*24/11/13*
Rédigé par :	**Tony Coriolle**
Relu par :	*Julien Szlamowicz, Delphine Meyrieux*

MISES A JOUR

Version	Date	Modifications réalisées
0.1	24/11/13	Création

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*

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*
- *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'intégrations :*

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 ».

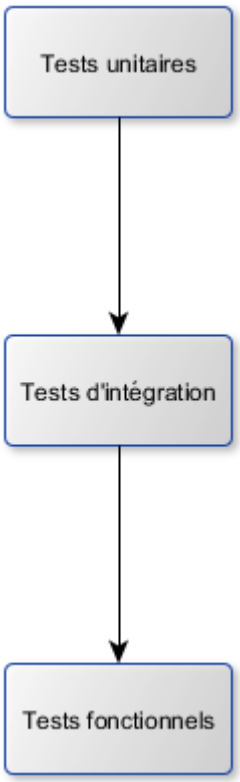
Le jeu de données sera composé de grands entiers qui devront être mis en place par nos soins. *« TeamCity », un outil d'intégration continue sur internet, nous permettra de déclarer des scénarios de tests, ainsi que de rassembler l'ensemble des tests unitaires des membres de l'équipe. Il nous permettra 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 technique vérifiera les tests existants et rajoutera 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.

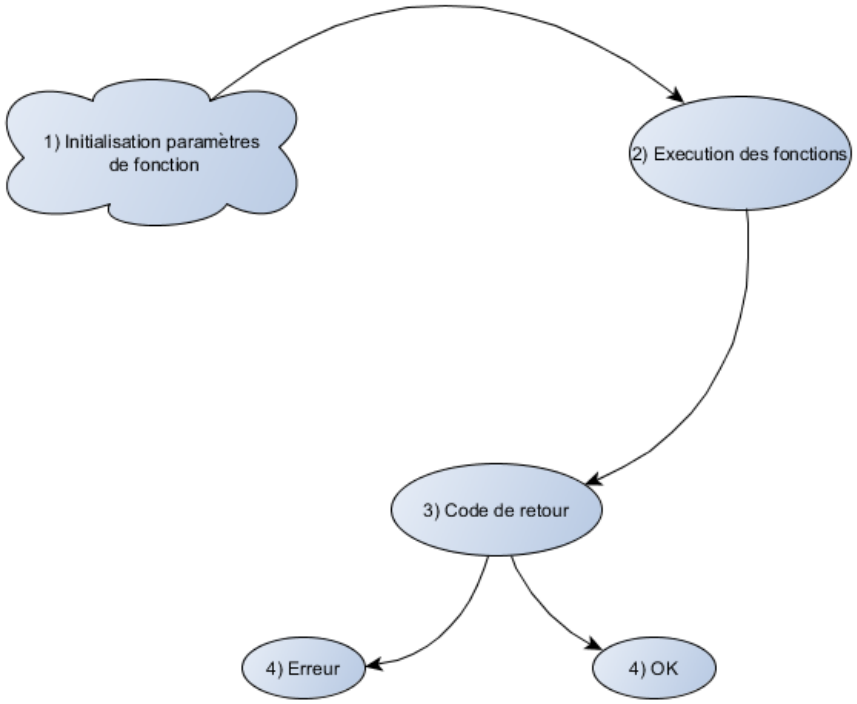
Le responsable technique devra automatiser au maximum les tests d'intégrations qui seront effectués via TeamCity, cela nous permettra d'avoir une intégration continue, d'identifier rapidement les parties de code défaillantes et de valider ou non les versions à mettre en recette.

6. Stratégie de tests

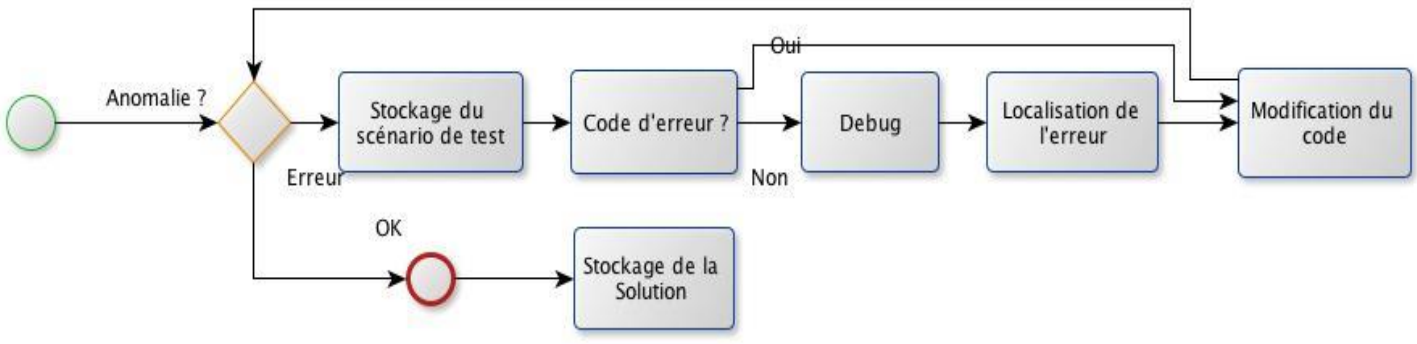


La campagne de test se déroulera de la façon suivante : Chaque fonctionnalité devra subir des tests unitaires ce qui lui permettra de pouvoir accéder à la deuxième étape des tests : l'intégration. Une fois les tests d'intégration réussis, le composant sera considéré comme valide et intégré au projet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

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. Puis en fonction du code de retour le test sera passé ou non.



7. Gestion des anomalies



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).

Lors de la rencontre d'anomalie le scénario de test devra être immédiatement stocké dans une base avec les paramètres entrés, l'anomalie survenue et quelle(s) solution(s) a été apportée(s) pour résoudre l'erreur.

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

8. Procédures de test

Test fonctionnel : Test de navigation		*Version : **0.1***		
Objectif du test : Menu				
Objet testé : Interface Graphique				
***Procédure n° [Test** UI – 1]*				
Précondition : aucune				
N°	***Actions***	***Résultats attendus***	***Conditions d'arrêt valide***	***Arrêt suite à erreur***
1	Cliquer sur les menus afin de se déplacer dans les sous menus	Obtenir le menu désiré lors du choix.		
2	Survoler la liste des algorithmes	Obtenir les informations désirées sur les algorithmes disponibles		
3	Saisir un nombre puis l'enregistrer	Nombre stocké en mémoire avec conversion si nécessaire		

Objet testé : ***SAGE***	*Version :* ***0.1***
Objectif de test : 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***
1	Déclencher l'initialisation de l'algorithme	Récupération du nombre et de la borne passés en entrée, conversion si nécessaire du nombre		
2	Exécution de l'algorithme	Produit de facteurs premiers égal à l'entier de départ	*Facteurs premiers*	*Facteurs premiers* *Erreur système*

Objet testé : ***SAGE***	*Version :* ***0.1***
Objectif de test : Heuristiques	
***Procédure n° [Test** SG – 2]*	
Précondition :	

N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	***Arrêt suite à erreur***
1	Déclencher l'initialisation de l'algorithme	Récupération du nombre et de la borne passés en entrée		
2	Exécution de l'algorithme	Produit de facteurs premiers égal à l'entier de départ		

Objet testé : ***CUDA***	*Version :* ***0.1***
Objectif de test : Algorithme de Dixon	
Procédure n° [Test CU - 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***
1	Déclencher l'initialisation de l'algorithme	Récupération du nombre et de la borne passés en entrée		
2	Exécution de l'algorithme	Produit de facteurs premiers égal à l'entier de départ	Facteurs premiers	Erreur système

Objet testé : ***CUDA***	*Version :* ***0.1***
Objectif de test : Heuristiques	
***Procédure n° [Test** CU – 2]*	
Précondition :	

N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	*Arrêt suite à erreur*
1	Déclencher l'initialisation de l'algorithme	Récupération du nombre et de la borne passés en entrée		
2	Exécution de l'algorithme	Produit de facteurs premiers égal à l'entier de départ		

Objet testé : ***Options***		*Version :* ***0.1***		
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***
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*
2	Génération XML	Rapport XML contenant les données de 1.	*Fichier généré*	

Objet testé : ***Options***		*Version :* ***0.1***		
Objectif de test : Comparaison 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***
1	Référencer les rapports à examiner	Stockage en mémoire des deux rapports		Fichiers non reconnus Format de fichier invalide Entiers différents
2	Lancer l'examen des rapports	Ce que le logiciel doit fournir comme résultat.	Rapport généré	

Objet testé : ***Options***			*Version :* ***0.1***	
Objectif de test : Pause/reprendre travail				
***Procédure n° [Test** MC – 3]*				
Précondition :				
N°	***Actions***	***Résultats attendus***	**Conditions d'arrêt valide**	***Arrêt suite à erreur***
1	Mettre en pause le programme	Arrêt des calculs		
2	Reprise du programme	Le programme reprend à l'endroit où il s'est arrêté		Perte de l'ordre des calculs Valeurs Incorrectes

Objet testé : ***Options***			*Version :* ***0.1***	
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***
1	Indiquer la base de représentation	Stockage en mémoire des deux rapports	Valeur correcte	Erreur de conversion