SPECIFICATION TECHNIQUE DE BESOIN

Version : **0.6**

Date : **30.03.2014**

Rédigé par : **Delphine Meyrieux**

Relu par : **Julien Szlamowicz, Tony Coriolle, Ibrahima Sory Barry, Clément Etendard, Timothée Guegan**

Approuvé par : **Christophe Carré**

Signature :

MISES A JOUR

Version	Date	Modifications réalisées
0.1	**14/11/2013**	**Création**
0.2	22/11/2013	Mise à jour
0.3	28/11/2013	Mise à jour
0.4	16/01/2014	Modification suite aux remarques du professeur de Gestion de projet
0.5	15/02/2014	Modification suite au redécoupage des tâches
0.6	16/03/2014	Modification suite au départ d'un membre de l'équipe

Ce document est destiné à traduire les besoins des utilisateurs du logiciel et à établir une référence pour sa validation. Son but est de recenser les principales exigences que l'équipe de développement s'engage à satisfaire dans le cadre du projet.

1. Objet :

L'objectif principal de ce projet est de développer un logiciel permettant de décomposer de grands nombres entiers en facteurs premiers grâce à l'algorithme de Dixon.

L'objectif technique est de proposer à l'utilisateur deux méthodes d'exécution par le biais de deux architectures pour en comparer les performances : Une méthode d'exécution **sérialisée** programmée en Sage et une seconde **parallélisée** programmée en C++. Cette dernière devra être exécutée en utilisant l'architecture CUDA. Il est pour cela nécessaire de maitriser entièrement CUDA, le projet portant essentiellement sur l'implémentation parallélisée.

Le but final de ce projet sera de comparer la vitesse d'exécution de l'algorithme entre les deux plateformes afin de démontrer que l'architecture CUDA permet d'effectuer des calculs complexes dans un temps beaucoup plus court.

2. Documents applicables et de référence :

Pour cette version, les documents de références sont :

- Le sujet du projet proposé par M. Carré.

3. Terminologie et sigles utilisés :

CUDA : fonctionnalité proposée sur les cartes graphiques récentes du constructeur NVIDIA, permettant d'utiliser la puissance de calcul de ces dernières pour paralléliser et exécuter des tâches extrêmement rapidement.

4. Exigences fonctionnelles :

1. Présentation des fonctionnalités du produit

Le projet s'articule en 4 parties distinctes, nous les différencierons de la manière suivante :

- Sage (SG)
- CUDA (CD)
- User Interface (UI)
- Miscellaneous (MC)

Ci-après, le tableau récapitulatif des fonctionnalités principales (backlog).

Tableau des fonctionnalités principales

Dom	Id	Intitulé	Priorité
SG	1	Algorithme de Dixon (Sage)	Indispensable
SG	2	Passerelle C++/Python	Important
SG	3	Heuristiques (Sage)	Facultatif
CD	1	Implémentation de l'algorithme du PGCD	Indispensable
CD	2	Fonction liste des nombres premiers	Indispensable
CD	3	Implémentation du remplissage de l'ensemble R	Indispensable
CD	4	Implémentation de la 3ème boucle	Indispensable
CD	5	Algorithme de Dixon (CUDA)	Indispensable
CD	6	Implémentation des Heuristiques (CUDA)	Facultatif
UI	1	Choix de la méthode d'exécution	Indispensable
UI	2	Choix de l'heuristique	Facultatif
UI	3	Choix du nombre à factoriser	Indispensable
UI	4	Description de l'algorithme	Facultatif
UI	5	Surveillance de l'exécution	Facultatif
UI	6	Affichage du rapport d'exécution	Important
MC	1	Générer un rapport au format *.xml*	Facultatif
MC	2	Comparaison de deux rapports	Facultatif
MC	3	Interruption/Reprise de l'exécution	Facultatif
MC	4	Conseil sur l'heuristique à utiliser	Facultatif
MC	5	Décimal, Hexa, Binaire	Important
MC	6	Etude sur l'algorithme de Dixon	Indispensable
MC	7	Optimisation du code	Important

2. Description des fonctionnalités

Dans la suite, la complexité sera pondérée en utilisant les premiers termes de la suite de Fibonacci pour qu'il n'y ait pas de valeurs trop similaires et ainsi pouvoir mettre en évidence les plus importants :

- Très Faible : 1
- Faible : 2
- Moyen : 3
- Elevé : 5
- Très Elevé : 8
- Critique : 13

<SG 1> Algorithme de Dixon (Sage).

CudaFactor

Module : Sage

Description de la fonctionnalité :

Implémenter l'algorithme de Dixon en Sage pour permettre à l'utilisateur de décomposer des nombres en facteurs premiers.
Entrée : nombre N, borne B
Sortie : Liste des facteurs premiers du nombre passé en entrée.
Début :
P <- ensemble des premiers < B
k <- card(P)
R <- ensemble vide
Div <- ensemble vide
Tester la primalité de N
Tant que $\prod_{i \in Div} Div_i$!= N faire
// élimination des facteurs triviaux
Pour x choisi aléatoirement dans [$\sqrt{N}$; N – 1] faire jusqu'à m > k
y ← x² mod N
Si y est B-friable et y pas dans Div alors
Ajouter (x,y) à R
m ← card(R)
Fin si
Finpour
Pour 1 ≤ i ≤ m faire
$v_{i,p}$ tq $y_i = \prod_{p \in P} p^{v_{i,P}}$
finpour
M ← matrice $(v_{i,p} \bmod 2)_{p \in P,\ 1 \le I \le m}$
E ← vecteur non nul noyau de M
u ← $\prod_{1 \le i \le m} x_i^{2e_i}$
v← $\prod_{p \in P} p^{\frac{1}{2}\sum_{1 \le i \le m} v_{i,p} e_i}$
si pgcd(u-v,N) n'est pas dans {1,N} alors
Ajouter u-v à Div
Sinon si pgcd(u+v,N) n'est pas dans {1,N} alors
Ajouter u+v à Div
Sinon
Erreur
finsi
Fintantque
Retourner DIv
Fin

Identificateur :

SG 1

Préconditions et Conditions de déclenchement/d'arrêt

N est un entier relatif.

Priorité :

Indispensable

Conditions d'acceptation :

Vérification et approbation du consultant technique (client).
Résultat correct : la suite des nombres retournée correspond bien aux facteurs premiers de N.
Temps d'exécution optimisé au maximum : $O(L_N(1/2; 2))$

Complexité :

13

<SG 2> Passerelle C++/Python (Sage).

CudaFactor	
Module : Sage	
Description de la fonctionnalité : Implémentation d'une passerelle entre C++ et Python afin de pouvoir lancer l'algorithme Sage depuis l'interface graphique de l'application.	Identificateur : SG 2
Préconditions et Conditions de déclenchement/d'arrêt Précondition : Avoir implémenté et vérifié la version générale de l'algorithme en Sage.	Priorité : Important
Conditions d'acceptation : Lance bien l'algorithme en Sage	Complexité : 8

<SG 3> Heuristiques (Sage).

CudaFactor	
Module : Sage	
Description de la fonctionnalité :	Identificateur :
Implémenter les heuristiques de l'algorithme de Dixon pour permettre à l'utilisateur d'adapter l'algorithme en fonction de ses besoins. Liste des heuristiques : • Crible quadratique (optimisé pour des nombres à moins de 100 chiffres) (QS) • Factorisation par crible sur les corps de nombres généralisé (GNFS) (optimisé pour des nombres à plus de 100 chiffres) • Fractions continues (CFRAC) Algorithme du crible quadratique : Notons $p_1, \dots, p_k$ les nombres premiers de la base des facteurs avec $K \approx \frac{\frac{1}{2}B}{\log B}$ On suppose T ≥ B. (1) Pour x = 0 à T, poser v[x] ←1 (2) Pour i = 1 à K, faire (a) Poser p ← p_i, e ← 1 (b) Faire R ← racines_de_ f(N, pe) (c) Si tous les éléments de R sont > T, passer à la prochaine valeur en 2 (d) Pour tout x ∈ R, Faire Tant que x ≤ T v[x] ← v[x] · p x ← x + p^e (e) Faire e ← e + 1 et retourner en 2.b (3) Renvoyer (x, f(x)) pour tous les x = 0, … , T tels que v[x] = f(x) Les deux autres algorithmes font partie de la recherche pour le moment.	SG 3
Préconditions et Conditions de déclenchement/d'arrêt	Priorité :
Précondition : Avoir implémenté et vérifié la version générale de l'algorithme en Sage.	Facultatif
Conditions d'acceptation :	Complexité :
Vérification et approbation du consultant technique (client) Temps d'exécution optimisé pour chaque heuristique $L_N(1/2,1)$. Résultat correct : la suite des nombres retournée correspond bien aux facteurs premiers de N.	8

<CD 1> Implémentation de l'algorithme du PGCD (CUDA).

CudaFactor	
Module : CUDA	
Description de la fonctionnalité : Implémentation parallélisée de l'algorithme du PGCD en C++ sur CUDA : Algorithme de décalage binaire.	Identificateur : CD 1
Préconditions et Conditions de déclenchement/d'arrêt Le nombre nb est bien un entier.	Priorité : Indispensable
Conditions d'acceptation : Résultat correct : le nombre renvoyé est le plus grand diviseur commun du nombre nb passé en argument.	Complexité : 8

<CD 2> Fonction liste des nombres premiers (CUDA).

CudaFactor	
Module : CUDA	
Description de la fonctionnalité : Implémentation parallélisée de la fonction permettant de construire la liste des nombres premiers inférieurs à la borne passée en argument en CUDA.	Identificateur : CD 2
Préconditions et Conditions de déclenchement/d'arrêt Le nombre borne passé en argument est bien un entier.	Priorité : Indispensable
Conditions d'acceptation : Résultat correct : liste de nombres premiers strictement inférieurs à la borne passé en argument.	Complexité : 8

<CD 3> Remplissage de l'ensemble R (CUDA).

CudaFactor	
Module : CUDA	
Description de la fonctionnalité : Implémentation parallélisée de la fonction permettant de remplir l'ensemble R contenant les couples (x,y) nécessaires au bon fonctionnement de l'algorithme. k = nombre de nombres premiers inférieurs à la borne tant que m < k + 1 faire choisir x aléatoirement entre $\sqrt{N}$ et $N - 1$ y = $x^2 mod\ N$ ajouter (x,y) dans R m = cardinal de R fin tant que	Identificateur : CD 3
Préconditions et Conditions de déclenchement/d'arrêt	Priorité : Indispensable
Conditions d'acceptation : Résultat correct : ensemble de couples (x,y) où y est le carré de x modulo N et le cardinal de cet ensemble est le même que celui de la liste des nombre premiers construite en <CD 2>	Complexité : 8

<CD 4> Fin de l'algorithme (CUDA).

CudaFactor	
Module : CUDA	
Description de la fonctionnalité : Implémentation parallélisée de la fonction de la fin de l'algorithme avec création de la matrice et construction des nombres u et v permettant de trouver un diviseur premier de nb. M <- matrice $(v_{i,p} \bmod 2)_{p\in P,\ 1<=i<=m}$ E <- vecteur non nul noyau de M u ← $\prod_{1 \leq i \leq m} x_i^{2e_i}$ v← $\prod_{p\in P} p^{\frac{1}{2}\sum_{1 \leq i \leq m} v_{i,p}e_i}$ si pgcd(u-v,N n'est pas dans {1,N} alors Ajouter u-v à Div Sinon si pgcd(u+v,N) n'est pas dans {1,N} alors Ajouter u+v à Div Sinon Erreur finsi	Identificateur : CD 4
Préconditions et Conditions de déclenchement/d'arrêt	Priorité : Indispensable
Conditions d'acceptation : Résultat correct : ensemble Div contenant les u+v et u-v facteurs premiers de nb.	Complexité : 13

<CD 5> Algorithme de Dixon : recomposition (CUDA).

CudaFactor

Module : CUDA

Description de la fonctionnalité :

Implémenter la reconstruction de l'algorithme suivant à partir des morceaux construits en <CD 1>, <CD 2>, <CD 3> et <CD 4>.

Entrée : nombre N, borne B
Sortie : Liste des facteurs premiers du nombre passé en entrée.
Début :
P <- ensemble des premiers < B
k <- card(P)
R <- ensemble vide
Div <- ensemble vide
Tester la primalité de N
Tant que $\prod_{i \in Div} Div_i$!= N faire
  // élimination des facteurs triviaux
  Pour x choisi aléatoirement dans [$\sqrt{N}$; N – 1] faire jusqu'à m > k
    Création d'un thread
    y <- x^2 mod N
    Si y est B-friable et y pas dans Div alors
      Ajouter (x,y) à R
      m <- card(R)
    Fin si
  finpour
  Pour 1 <= i <= m faire
    $v_{i,p}$ tq $y_i = \prod_{p \in P} p^{v_{i,P}}$
  finpour
  M <- matrice $(v_{i,p} \bmod 2)_{p \in P,\ 1<=i<=m}$
  E <- vecteur non nul noyau de M
  u ← $\prod_{1 \leq i \leq m} x_i^{2e_i}$
  v← $\prod_{p \in P} p^{\frac{1}{2}\sum_{1 \leq i \leq m} v_{i,p} e_i}$
  si pgcd(u-v,N n'est pas dans {1,N} alors
    Ajouter u-v à Div
  Sinon si pgcd(u+v,N) n'est pas dans {1,N} alors
    Ajouter u+v à Div
  Sinon
    Erreur
  finsi
Fintantque
Retourner Div
Fin

Identificateur :

CD 5

Préconditions et Conditions de déclenchement/d'arrêt

N est un entier relatif.

Priorité :

Indispensable

Conditions d'acceptation :

Résultat correct satisfaisant après soumission au client. Temps d'exécution optimisé au maximum : $O(L_N(1/2; 2))$
Résultat correct : la suite des nombres retournée correspond bien aux facteurs premiers de N.

Complexité :

13

<CD 6> Heuristiques (CUDA).

CudaFactor	
Module : CUDA	
Description de la fonctionnalité : Implémenter et paralléliser les heuristiques de l'algorithme de Dixon en C++ sur CUDA pour permettre à l'utilisateur de décomposer des nombres en facteurs premiers. Liste des heuristiques : • Crible quadratique (optimisé pour des nombres à moins de 100 chiffres) (QS) • Factorisation par crible sur les corps de nombres généralisé (GNFS) (optimisé pour des nombres à plus de 100 chiffres) : utilisant de la théorie algébrique des nombres et reposant sur la factorisation d'idéaux dans des anneaux d'entiers • Fractions continues (CFRAC) Algorithme du crible quadratique : cf. SG2	Identificateur : CD 6
Préconditions et Conditions de déclenchement/d'arrêt Précondition : Avoir implémenté et vérifié la version générale de l'algorithme pour CUDA.	Priorité : Facultatif
Conditions d'acceptation : Vérification et approbation du consultant technique (client) Temps d'exécution optimisé pour chaque heuristique $L_N(1/2,1)$. Résultat correct : la suite des nombres retournée correspond bien aux facteurs premiers de N.	Complexité : 8

<UI 1> Choix de la méthode d'exécution.

CudaFactor	
Module : User Interface	
Description de la fonctionnalité : L'utilisateur doit pouvoir choisir la méthode d'exécution de l'algorithme : Sage ou CUDA.	Identificateur : UI 1
Préconditions et Conditions de déclenchement/d'arrêt NA.	Priorité : Indispensable
Conditions d'acceptation : L'action enregistre le choix de la méthode à exécuter et mène à l'exécution de la bonne méthode.	Complexité : 2

<UI 2> Choix de l'heuristique.

CudaFactor	
Module : User Interface	
Description de la fonctionnalité : L'utilisateur doit pouvoir choisir l'heuristique de l'algorithme de Dixon qu'il souhaite utiliser.	Identificateur : UI 2
Préconditions et Conditions de déclenchement/d'arrêt NA.	Priorité : Facultatif
Conditions d'acceptation : L'action enregistre bien l'heuristique choisie et mène à l'exécution du bon algorithme.	Complexité : 2

<UI 3> Choix du nombre à factoriser.

CudaFactor	
Module : User Interface	
Description de la fonctionnalité : L'utilisateur doit pouvoir choisir le nombre à factoriser.	Identificateur : UI 3
Préconditions et Conditions de déclenchement/d'arrêt Une méthode d'exécution et un algorithme ont été choisis.	Priorité : Indispensable
Conditions d'acceptation : L'action enregistre le nombre à factoriser et lance la factorisation.	Complexité : 3

<UI 4> Description de l'algorithme.

CudaFactor	
Module : User Interface	
Description de la fonctionnalité : L'utilisateur doit pouvoir avoir une description rapide de l'algorithme (ou de l'heuristique) qu'il s'apprête à utiliser pour être sûr qu'il fait un choix cohérent.	Identificateur : UI 4
Préconditions et Conditions de déclenchement/d'arrêt Condition de déclenchement : L'utilisateur présélectionne une heuristique. Condition d'arrêt : L'utilisateur effectue une autre action.	Priorité : Facultatif
Conditions d'acceptation : Description correcte et synthétique.	Complexité : 2

<UI 5> Surveillance de l'exécution.

CudaFactor	
Module : User Interface	
Description de la fonctionnalité : L'utilisateur peut connaitre l'état d'avancement des calculs, le nombre d'instructions par seconde ou l'état général du système en temps réel et des détails comme combien de colonnes de la matrice a été trouvé dans la phase 1 de l'algorithme de Dixon.	Identificateur : UI 5
Préconditions et Conditions de déclenchement/d'arrêt Condition de déclenchement : L'utilisateur lance une factorisation. Condition d'arrêt : Le nombre a été factorisé.	Priorité : Facultatif
Conditions d'acceptation : Les valeurs rapportées sont exactes et s'actualisent en temps réel.	Complexité : 5

<UI 6> Affichage du rapport d'exécution.

CudaFactor	
Module : User Interface	
Description de la fonctionnalité : L'utilisateur peut avoir un compte rendu de l'exécution à la fin de celle-ci. Le rapport est composé de 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.	Identificateur : UI 6
Préconditions et Conditions de déclenchement/d'arrêt Condition de déclenchement : La factorisation est terminée.	Priorité : Important
Conditions d'acceptation : Les valeurs doivent être celles rendues par l'algorithme. Présentation claire.	Complexité : 5

<MC 1> Générer un rapport au format *.xml*.

CudaFactor	
Module : Miscellaneous	
Description de la fonctionnalité : L'utilisateur peut générer un rapport d'exécution au format *.xml*.	Identificateur : MC 1
Préconditions et Conditions de déclenchement/d'arrêt Condition de déclenchement : La factorisation est terminée.	Priorité : Facultatif
Conditions d'acceptation : Pas d'erreur de syntaxe dans le langage. Hérite des caractéristiques de l'UI 6.	Complexité : 5

<MC 2> Comparaison de deux rapports.

CudaFactor	
Module : Miscellaneous	
Description de la fonctionnalité : L'utilisateur peut comparer deux rapports de précédentes exécutions, sous forme de tableau avec une coloration des données : rouge pour les moins performantes et vert pour les plus performantes.	Identificateur : MC 2
Préconditions et Conditions de déclenchement/d'arrêt Précondition : les rapports concernent une factorisation du même nombre, les rapports sont au format .xml	Priorité : Facultatif
Conditions d'acceptation : Comparatif synthétique et correct des deux rapports. Mise en avant des données performantes.	Complexité : 5

<MC 3> Interruption/Reprise de l'exécution.

CudaFactor	
Module : Miscellaneous	
Description de la fonctionnalité : L'utilisateur peut mettre en pause la factorisation pour la reprendre plus tard, ou simplement l'interrompre.	Identificateur : MC 3
Préconditions et Conditions de déclenchement/d'arrêt NA.	Priorité : Facultatif
Conditions d'acceptation : Le programme libère correctement les ressources.	Complexité : 8

<MC 4> Conseil sur l'heuristique à utiliser.

CudaFactor	
Module : Miscellaneous	
Description de la fonctionnalité : L'utilisateur est conseillé sur l'heuristique à utiliser en fonction de l'entier qu'il souhaite factoriser. (Certaines heuristiques sont plus efficaces pour certains types de nombres)	Identificateur : MC 4
Préconditions et Conditions de déclenchement/d'arrêt Condition de déclenchement : L'utilisateur a choisi un nombre à factoriser.	Priorité : Facultatif
Conditions d'acceptation : Conseil approprié à la taille de l'entier.	Complexité : 2

<MC 5> Décimal, Hexa, Binaire.

CudaFactor	
Module : Miscellaneous	
Description de la fonctionnalité : L'utilisateur peut entrer le nombre à factoriser dans plusieurs bases : décimale, hexadécimale, binaire car il peut être amené à entrer de très grands nombres. Conversion des nombres entrés dans l'application.	Identificateur : MC 5
Préconditions et Conditions de déclenchement/d'arrêt La base choisie doit être soit décimale, hexadécimale ou binaire.	Priorité : Important
Conditions d'acceptation : Le nombre entré et le nombre converti sont égaux.	Complexité : 5

<MC 6> Etude sur l'algorithme de Dixon

CudaFactor	
Module : Miscellaneous	
Description de la fonctionnalité : Réaliser une étude préliminaire des algorithmes et heuristiques nécessaires au développement de l'application.	Identificateur : MC 6
Préconditions et Conditions de déclenchement/d'arrêt NA.	Priorité : Indispensable
Conditions d'acceptation : NA.	Complexité : 8

<MC 7> Optimisation du code

CudaFactor	
Module : Miscellaneous	
Description de la fonctionnalité : Apporter des corrections et optimisations sur les implémentations des algorithmes en SAGE et CUDA.	Identificateur : MC 7
Préconditions et Conditions de déclenchement/d'arrêt Précondition : Avoir implémenté et vérifié la version générale et les heuristiques de l'algorithme.	Priorité : Important
Conditions d'acceptation : Résultat satisfaisant après soumission au client. Code correct.	Complexité : 13

5. Exigences opérationnelles :

Nom	Identificateur	Priorité
Une carte graphique NVIDIA supportant CUDA nous permettant le développement en CUDA	EO 1	Indispensable
Un système d'exploitation UNIX nous permettant le développement du projet ainsi que son exécution	EO 2	Indispensable

Ces exigences sont nécessaires à la réalisation de toutes les exigences fonctionnelles citées précédemment.

6. Exigences de qualité

Nom	Identificateur	Priorité
Exécution rapide de l'algorithme en SAGE	EQ 1	Indispensable
Exécution de l'algorithme en CUDA optimisée par rapport au temps d'exécution obtenu avec SAGE ce qui dépend de la parallélisation que l'on fera sur celui-ci	EQ 2	Indispensable

7. Exigences d'interface

L'interface n'étant pas une priorité pour la réussite du projet, nos exigences seront les suivantes :

Nom	Identificateur	Priorité
Simple	EI 1	Important
Fonctionnelle	EI 2	Important
Fluide	EI 3	Important
Ergonomique	EI 4	Important

8. Cas d'utilisation