PLAN DE DEVELOPPEMENT

Version :	*1.0*
Date :	*28.05.2014*
Rédigé par :	*Clément Etendard*
Relu par :	*Julien Szlamowicz, Delphine Meyrieux, Tony Coriolle, Timothée Guegan*

MISES A JOUR

Version	Date	Modifications réalisées
0.1	29/11/2013	Création
0.2	16/01/2014	Modifications suite aux remarques du professeur de Gestion de Projet
0.3	28/03/2014	Modifications suite à l'abandon d'un développeur
1.0	28/05/2014	Version finale

1. Contexte du projet

- Ce projet universitaire a pour but de réaliser un logiciel permettant de décomposer de grands nombres entiers en ***facteurs premiers*** grâce à l'algorithme de ***Dixon***. Cet algorithme est utilisé notamment dans les systèmes cryptographiques car il répond à un problème mathématique complexe. Mais le temps d'exécution de cet algorithme est-il plus rapide sous ***Sage*** ou sous ***CUDA*** ?
  - Sage est une surcouche du langage Python spécialisée dans le calcul formel. Il effectue les calculs de façon linéaire en utilisant la puissance du CPU.
  - CUDA est le langage de programmation des ***cartes graphiques*** du constructeur ***Nvidia***, permettant de tirer profit de la puissance de calcul phénoménale dont elles disposent. En effet, les cartes graphiques disposent jusqu'à des milliers de micro-processeurs dédiés uniquement aux calculs élémentaires nécessaires notamment pour la 3D.

C'est pourquoi nous réaliserons une application exécutant l'algorithme de Dixon sur ces deux architectures et constaterons laquelle est le plus rapide.

- L'équipe sera encadrée par le client : *Mr. Christophe Carré – Maitre de conférences au pôle Science et Techniques de l'Université de Rouen*-, qui validera nos états d'avancement au fur et à mesure du projet. Le sujet évoluant dans son domaine de compétences, il sera également à même de nous aider notamment pour les questions théoriques, pendant la durée du développement.

Le projet débutera après les examens du premier semestre. Nous suivrons le planning développé lors du module "Gestion de Projet", que vous pouvez consulter dans la section 6.

Les contraintes de développement sont l'utilisation de Sage, une optimisation de l'algorithme de Dixon avec CUDA, et une réalisation finale de qualité. Les rapports des résultats d'exécution doivent respecter le format XML. Nous nous sommes imposé ce format, afin de rendre les résultats des rapports d'exécution plus faciles à comparer et à exporter.

Nous développerons principalement la partie CUDA en groupe, dans la salle informatique de l'université mise à notre disposition, mais également chacun chez soi. Les tests sur la partie CUDA seront réalisés dans cette salle informatique car une machine équipée d'une carte graphique Nvidia y a été mise à notre disposition.

Client :

- Christophe Carré

Équipe technique :

- (JS) Julien Szlamowicz (Chef de Projet, Développeur et Testeur)
- (CE) Clément Etendard (Responsable organisation, Développeur et testeur)
- (DM) Delphine Meyrieux (Responsable Client, Développeur et Testeur)
- (TC) Tony Coriolle (Responsable Qualité, Développeur et Testeur)
- (TG) Timothée Guegan (Responsable Technique, Développeur et testeur)

- Les Objectifs :

  - Mener une étude sérieuse, complète, et compréhensible pour tout type d'utilisateur (documents soignés, explications détaillées, tableau comparatif simple et efficace).
  - Retour de résultats esthétiques, performants et de qualité.
  - Rendre le travail dans le temps imparti.
  - Avancer le projet sous forme d'itérations livrables, c'est à dire développer en premier lieu un noyau minimal fonctionnel destiné à accueillir de manière itérative des fonctionnalités.

Enfin, être soudé les uns les autres, comprendre, étudier le problème et développer le logiciel tous ensemble afin de répondre au mieux au besoin exprimé par le client.

- Documents utilisés :
  - J.Sanders, E.Kandrot, *Cuda par l'exemple, Une introduction à la programmation parallèle de GPU*, 2011.
  - D. Vergnaud, *Exercices et problèmes de cryptographie*, Dunod, 2012.
  - Stinson, Chapman and Hall, *Cryptographie, Theory and Practice,* Third Edition, 2006.
  - J. von zur Gathen, J.Gerhard, *Modern Computer Algebra*, Sd Edition, 2003, Cambridge University Press.

2. Méthodologie de développement

Nous avons tout d'abord réalisé une étude du sujet afin de mettre en évidence les tâches principales pour les stocker dans le backlog.

Nous avons d'abord étudié les méthodes agiles, très utile aux entreprises pour développer des logiciels de bonne qualité rapidement et efficacement. Après cette étude, notre choix s'est porté sur la méthode Scrum, enrichie avec un tableau de type Kanban (utilisé principalement dans l'industrie et dans la méthode Lean)

- Scrum

  Nous utiliserons principalement cette méthode notamment pour :

  - l'organisation :

    Un scrum master, un product owner (le client) et des développeurs.
    Le système de réunion courte et régulière pour ne pas perdre la communication.

  - le découpage du travail en user stories :

    Qui permet d'obtenir des résultats concrets rapidement en accord avec le client qui choisit les prochaines fonctionnalités du backlog à implémenter pour la prochaine itération.

- Kanban

  Un tableau (normalement physique) sur lequel on déplace les tâches et qui permet donc, en plus des réunions régulières, de savoir sur quoi travaille chacun et d'éviter l'effet d'isolement (par exemple deux personnes qui travaillent de manière concurrente sur une même fonctionnalité).

Nous utiliserons Youtrack, un outil en ligne dédié à la gestion de projet agile, qui nous permet d'avoir un tableau de type Kanban et d'y stocker un backlog. Quand un membre de l'équipe a fini de développer et a validé une fonctionnalité, il s'en attribue une nouvelle dans la liste de celles qu'il reste à traiter et dans son domaine de compétence (cf. Organigramme des tâches).

Cet outil nous permet aussi d'afficher le graphique du backlog pour le sprint en cours et pour la durée totale du projet et de nous situer par rapport à la courbe affine optimale de développement.

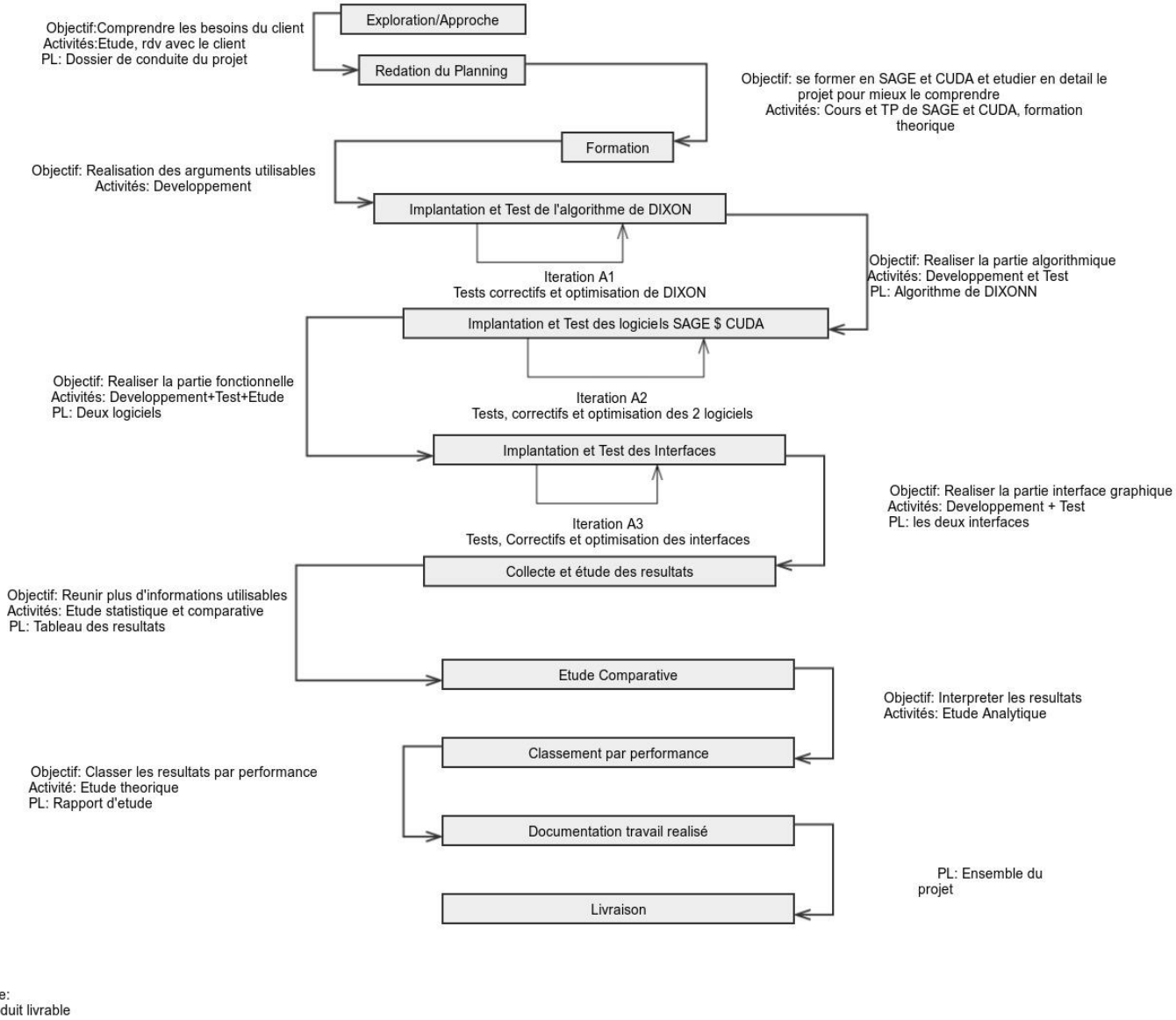
De même, on pourra éditer toutes sortes de graphiques tels que la somme totale des complexités des fonctionnalités développées par un membre de l'équipe sur un temps ou un sprint donné (ce qui nous permettra d'identifier les blocages).

Enfin, cet outil s'interface aussi avec d'autres outils que nous utiliserons pour le développement tel que Github pour le stockage et TeamCity pour l'intégration continue et les tests unitaires.

Nous avons choisi GitHub pour stocker le code, car il permet une bonne gestion des versions des fichiers sur le serveur, même si deux personnes travaillent sur le même fichier. Les annulations de soumissions sont rapides et il permet au client de pouvoir regarder et commenter le code sans le modifier.

Notre méthode, des valeurs à respecter :

- Satisfaction du client de bout en bout du projet, respecter le cahier des charges imposé, écouter ses revendications, appliquer ses recommandations, et l'aiguiller également vers des optimisations que nous pourrions juger utiles.
- Être capable de modifier rapidement le code, les techniques utilisées (avoir le contrôle sur notre projet)
- Livrer les avancements du projet dès que possible (pour rassurer le client sur la qualité du produit, et l'équipe sur le bon déroulement des opérations)
- Le client a accès continuellement à l'avancement du projet grâce à un compte qui lui est dédié sur Youtrack.
- L'équipe doit être motivée, apprendre en s'intéressant au problème à étudier pour coder efficacement avec des bases du projet solides, ainsi que de soutenir le travail de chacun.
- Communiquer avec les autres acteurs (client, développeurs, testeurs, ...) en priorité, lors de réunions ou mises au point, puis secondairement à l'aide d'une messagerie (boite de messagerie personnelle et de l'université, ...).
- Se concentrer sur l'objectif principal, choisir la simplicité dans la gestion du développement (pas sur le développement lui-même), éviter le travail inutile.
- Communiquer de façon claire et publique (mettre toute l'équipe en pièce jointe) même si l'on pense que ce n'est pas intéressant, de sorte que tous les membres de l'équipe disposent des mêmes informations.



Nous présentons le schéma de notre méthodologie de projet, sous forme de diagramme UML : il décrit les objectifs, les activités et les produits livrables de chaque étape de développement.

Les flèches indiquent le sens de circulation, l'étape initiale étant : 'Exploration/Approche' et l'étape finale étant : le livrable.

A titre d'exemple de la première étape 'Exploration/Approche' à la seconde étape 'Rédaction du Planning', l'objectif est la compréhension du projet, l'activité réalisée est l'étude et réalisation des dossiers de conduite du projet et le produit livrable (PL) est le dossier de conduite de projet.

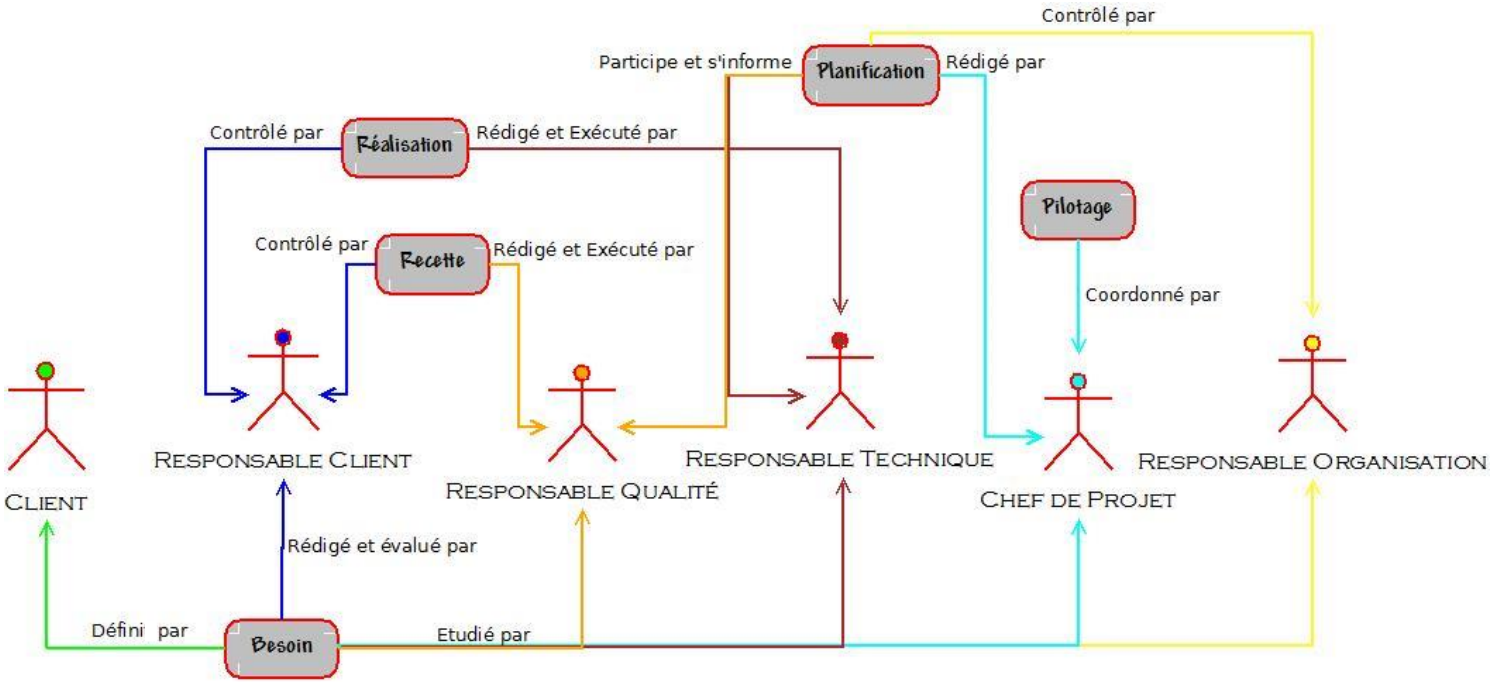
Les Itérations A.1, A.2 et A.3 sont des boucles qui permettent de répéter les différentes itérations afin de s'assurer que tous les tests, correctifs et optimisations soient effectués sur les algorithmes et logiciels avant de passer à l'étape suivante.

3. Organisation et responsabilités

Le projet est organisé par une équipe technique composée de 5 personnes. Chaque personne est à la fois développeur et testeur, et certaines ont également une spécialité. Cette équipe est composée :

- d'un chef de projet : Julien Szlamowicz
- d'un responsable organisation : Clément Étendard
- d'un responsable client : Delphine Meyrieux
- d'un responsable qualité : Tony Coriolle
- d'un responsable technique : Timothée Guegan

Le rôle de chaque spécialité est défini dans le schéma joint ci-dessous.



Chef de Projet : Distribue, contrôle et valide les tâches à effectuer, fait également le lien entre tous les autres rôles. Doit être un soutien pour tous.

Responsable Client : Présente l'avancement du projet et recueille les demandes (mise à jour du sujet, nouvelles technologies à utiliser...), les mécontentements, et les remarques du client. Prend également en main, la mise en forme du projet, pour ne pas le voir dévier de son objectif final. Il est le garant de la vision client.

Responsable Qualité : Vérifie les fonctionnalités des logiciels, des algorithmes utilisés, et des arguments utilisés. En cas de mise à jour, ou d'optimisation il doit réitérer les tests.
En cas d'erreur, de bug ou de mauvaise appréciation, il en fait part aux autres, et une solution doit être trouvé le plus rapidement possible.

Responsable Organisation : Vérifie que chaque développeur ne prend qu'une tâche à la fois, que celui-ci ne reste pas bloqué sur une tâche pendant un temps anormalement long et que le logiciel ne prend pas de retard. Il est le bras droit du Chef de Projet.

Responsable Technique : Il est responsable de l'architecture du produit, c'est lui qui doit trancher et prendre les décisions sur les choix d'implémentation. Il doit également conseiller et coordonner la production et le développement technique du logiciel.

4. Organigramme des tâches

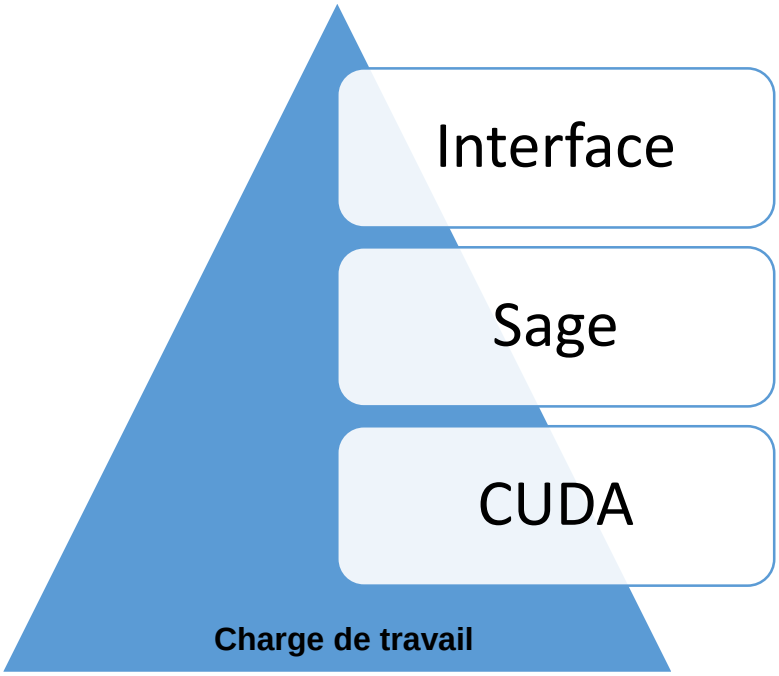
Etant donnée l'architecture un peu spéciale que décrit notre projet, nous avons décidé d'adopter une stratégie de développement particulière. En effet, notre sujet se divise en 3 axes principaux que sont :

- L'interface graphique (IHM) :
  - Choix de la méthode d'exécution
  - Choix du format du nombre à factoriser (Décimal, Hexa, Binaire)
  - Choix du nombre à factoriser
  - Surveillance de l'exécution
  - Affichage du rapport d'exécution
- La partie Sage (SG) :
  - Etude de l'algorithme de Dixon
  - Algorithme de Dixon (Sage)
- La partie CUDA (CD) :
  - Etude de l'algorithme de Dixon
  - Algorithme de Dixon (CUDA)
  - Optimisation du Code

Dans un souci d'efficacité, nous avons réparti les équipes pour que le travail soit partagé en 3 sous équipes qui s'occuperont respectivement d'une partie distincte.

La charge de travail sur les trois parties étant croissante, notre stratégie sera la suivante :

Chaque groupe aura pour but de produire des documents synthétisés sur le sujet qu'il a étudié pour permettre une rapide montée en compétence du reste de l'équipe quand ce sera nécessaire.

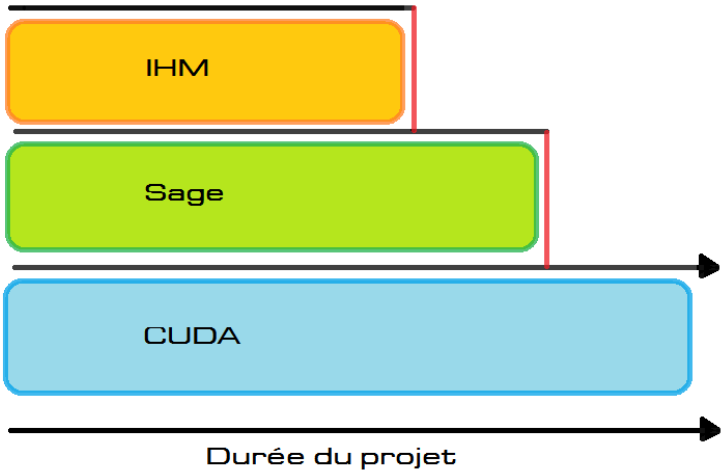


Comme on peut le constater, l'interface nécessite un travail moins important que le Sage qui sera lui-même inférieur à CUDA.

Ainsi, le groupe qui travaillera sur l'interface graphique au lancement de la phase de développement rejoindra rapidement celui qui sera sur la problématique Sage. Il faudra donc une rapide montée en compétence pour pouvoir s'intégrer à la partie Sage.
Une fois la partie Sage terminée, l'effectif passera à plein temps sur la problématique CUDA, qui est l'aspect technique majeur du projet.
Les documents produits pendant la phase de recherche permettront au reste de l'équipe de gagner du temps sur la montée en compétence et de se dispenser de recherches fastidieuses et chronophages qui auront déjà été réalisées par les groupes en amont.

Les groupes de veilles seront :

- Interface :
  - Clément Etendard (CE)
- Sage :
  - Delphine Meyrieux (DM)
  - Julien Szlamowicz (JS)
- CUDA :
  - Tony Coriolle (TC)
  - Timothée Guegan (TG)



Sur la figure ci-avant, les flèches noires représentent la phase de développement des sous équipes et les flèches rouges, la montée en compétence accélérée.

La partie CUDA étant le cœur du projet, il est évident qu'elle soit la plus conséquente. De même, elle nécessitera des optimisations constantes et est donc la moins quantifiable, C'est pourquoi nous lui accorderons le plus de temps et de ressources.

Cette stratégie nous permettra de minimiser les risques de blocages et donc de gagner du temps dans le développement.

5. Evaluation du projet et dimensionnement des moyens

- *Temps moyen de réalisation :*

Au total, le projet contient 150 jours de travail. Comme décrit précédemment, le projet est divisé en trois parties bien distinctes. La première partie est la réalisation de l'interface graphique, pour laquelle nous travaillerons pendant 15 jours (10%). La seconde est la programmation Sage, pour laquelle nous estimons 37 jours (25%) de travail. Puis la dernière partie est la programmation CUDA, pour laquelle nous prendrons 98 jours (65%) de travail. Chaque partie est réalisée en binôme, puis lorsqu'un binôme a fini sa partie, il passe à la partie suivante. Comme décrit dans la partie précédente.

Nous ne comptabilisons ni les réunions, ni le temps d'apprentissage et de maîtrise des outils, ni les entretiens avec le client.

Ce temps est donné à titre indicatif, il doit simplement servir de base à l'avancement du projet, le planning sera amené à évoluer.

Notre sujet est assez particulier, la notion de fin est ambigüe, puisque l'on peut toujours essayer de réduire le temps d'exécution de l'algorithme et dépend aussi du matériel sur lequel il est exécuté.

Nous définirons les jours hommes tels que : 1h40/jour du lundi au vendredi.
Une semaine est donc composée de 5 jours hommes soit 8heures de travail par personne.

Le projet débutera sa phase de développement le 20 Janvier 2014, après que les documents finaux soient validés.

Et le contrat devra être rempli aux alentours du 9 Mai 2014.

- Le besoin en moyens et en ressources :
  - *Pour la plate-forme de développement (matériels, logiciels et outils):*
    - *Un ordinateur personnel chacun, en plus de ceux des salles de TP.*
    - *Codage principal sur la machine mise à notre disposition utilisant la puissance de la carte graphique Nvidia.*
    - *Utilisation du logiciel Sage pour le développement sur Sage et de GanttProject*
    - *Utilisation des langages C, C++, XML, CUDA.*

6. Planning général

La durée des Sprints pour le projet est de 1 mois (premier sprint : du 20 janvier au 18 février 2014), il y aura donc chaque mois une réunion avec le client et un livrable. En début de sprint, l'équipe décidera des fonctionnalités à implémenter pendant la première moitié du sprint.

L'équipe se réunira ensuite à la moitié du sprint pour faire le point sur les fonctionnalités qu'il reste à développer pour l'itération en cours.

Le déroulement d'un Sprint est décrit dans le planning de Gantt fourni en annexe.
Comme rappelé précédemment, la structure pyramidale de notre projet impliquera que les plannings des sprints diffèreront grandement du premier au fur et à mesure que le projet avancera car les sous équipes seront amenées à évoluer.

Les sprints étant continus, une longue réunion sera nécessaire à la fin de chaque sprint (donc au commencement du suivant) et condensera :

- Le **sprint review** : L'objectif de cette partie est de valider le logiciel produit pendant le sprint auprès du client. Pour cela, l'équipe ou chaque sous équipe effectue une démonstration du logiciel produit devant le client. C'est sur la base de cette démonstration que le client valide ou non chaque fonctionnalité planifiée pour ce sprint.

- Le **sprint planning** : L'objectif de cette partie est de déterminer avec le client le but du prochain sprint, c'est-à-dire les fonctionnalités restantes dans le backlog produit qui devront être déplacées dans le backlog du sprint et ainsi être développées pour la prochaine itération.

- La **rétrospective du sprint** : La présence du client à cette partie de la réunion n'est plus nécessaire, l'objectif de cette dernière est de comprendre ce qui n'a pas bien marché pendant le sprint, les erreurs commises et de prendre des décisions pour s'améliorer. Il est tout à fait possible d'apporter des aménagements à la méthode Scrum. Pendant cette phase, chaque membre de l'équipe dispose de post-it sur lesquels il inscrit ce qu'il a apprécié ou, au contraire, ce qui lui a déplu en pondérant ses remarques tels que : (- -) = très mauvais , (-) = mauvais, (+) = bon, (+ +) = très bon. Ensuite, chacun leurs tours, les membres de l'équipe se lèvent pour apposer leurs remarques au mur en les commentant. On discute ensuite des solutions pour pallier les éventuels problèmes.

7. Procédés de gestion

7.1. Gestion de la documentation

Pendant le projet, les documents à produire seront :

- Un dossier contenant des tableaux de résultats comparatifs, compréhensibles et précis. Qui a pour but de répondre à la problématique du projet à l'aide des données recueillies.
- Un document rassemblant toutes nos recherches sur l'algorithme de Dixon et les langages utilisés.
- Un document de suivi d'optimisation de l'algorithme en CUDA.
- Un manuel d'utilisation.

Tout sera rédigé communément (la plus grosse partie de notre projet étant l'étude), chacun doit apporter ses idées, et ses résultats selon les tâches confiées.

7.2. Gestion des configurations

Organisation des espaces :

Principalement à l'université, lors de réunions (pour le partage des tâches, les risques d'indisponibilités de chacun, le questionnement, ou la mauvaise interprétation).

Mais aussi chez soi, pour la partie théorique.

Sauvegardes et archivages :

Nous avons choisi, comme moyen de stockage partagé la plateforme en ligne GitHub qui permet un accès concurrent aux données présentes sur le serveur. Ainsi, chacun des développeurs peut travailler sur un fichier sans craindre d'écraser le travail de son collègue et permet un avancement et une sécurité maximale.

De plus cette plateforme permet de prendre en charge différentes versions d'un fichier. Le commit se fait régulièrement en local puis on « push » les données qu'une seule fois sur le serveur. Ce qui permet d'annuler facilement et rapidement des modifications apportées en cas de maladresse. Il permet une gestion des branches de bonne qualité car une branche permet de bosser sur différents sous projets sans avoir à déposer le tout pour que n'importe qui le modifie.

Traitement des évolutions :

Les tâches à faire seront mises à disposition sur Youtrack par le chef de projet en accord avec le client et le responsable client, quand un développeur a fini une tâche, il en reprend une nouvelle en indiquant sur Youtrack qu'il travaille sur cette tâche. Le responsable Organisation doit vérifier qu'un développeur ne s'est pas attribué plus d'une tâche à la fois. Ce système permet à chacun de savoir à tout instant sur quoi travaillent les autres et au chef de projet d'identifier les blocages pour y remédier au plus vite.

Le chef de projet a pour rôle de gérer les configurations du projet, mais tout le monde est actif sur ce principe, une communication quotidienne sur les étapes du projet est primordiale (en cas de blocage sur une tâche par exemple).

8. Revues et points clefs

Les principaux objectifs du projet sont :

- L'étude de l'algorithme de Dixon
- L'optimisation de l'algorithme de Dixon en CUDA
- Présentation de résultats clairs et cohérents

Le calendrier prévisionnel est décrit dans la section 4 : l'Organigramme des tâches.

A la fin de chaque sprint, un livrable est soumis au client, ce qui déclenche une réunion de début de sprint afin de fixer les objectifs de celui-ci.

9. Procédure de suivi d'avancement

Définir les procédures mises en œuvre pendant le projet pour en assurer le suivi :

- *Selon l'emploi du temps du second semestre, nous organiserons un agenda en phase avec notre planning. Le processus sera de suivre les étapes de cet agenda, et en cas de non-délai respecté, en informer les autres pour pouvoir terminer cette tâche au plus vite. Le rôle du chef de projet et de tenir le projet en route, et prévoir les obstacles à l'avance. Fréquence : Quotidienne, sans prendre la journée pour suivre l'avancement, quelques minutes pour définir le topo, et mettre tout le monde d'accord.*

- Lors de l'avancement du projet, tout le monde peut voir la nouvelle tâche finie sur Youtrack puis avoir accès au code sur GitHub. Le client peut également voir l'avancement du projet sur Youtrack.

- Une longue réunion mensuelle avec le client sera organisée pour clôturer la fin du sprint en cours et en débuter un nouveau.

- *Une réunion courte quotidienne d'équipe de type SUM (Stand Up Meeting) :*

    - *Chaque membre prend la parole à tour de rôle et réponds à 3 questions simples :*
        - *Qu'ai-je fait hier ?*
        - *Que vais-je faire aujourd'hui ?*
        - *Ai-je des remarques, des problèmes ou des blocages quels qu'ils soient ?*