ARCHITECTURE DU LOGICIEL

Version :	1
Date :	10/01/2012
Rédigé par :	Emmanuel Mocquet
Relu par :	Baptiste Dolbeau
Approuvé par :	

MISES A JOUR

Version	Date	Modifications réalisées
0.1	16/12/11	Création
0.2	10/01/12	Complétion et modification
1	16/01/12	Suppression d'un module inutilisé

1. Objet :

Ce document a pour but de décrire les solutions techniques conçues pour répondre aux exigences définies dans la spécification technique de besoin. Il doit identifier et décrire les différents modules ou constituants du logiciel ainsi que leurs interfaces de telle sorte que chacun d'entre eux puisse être développé de façon autonome par un membre de l'équipe avant d'être intégré.

Ce logiciel doit simuler un périphérique, composé d'un SSD et d'un disque dur mécanique, et créer un driver contrôlant ce dernier.

La lecture de ce document doit également permettre d'appréhender l'ensemble des paramètres techniques pris en considération par les auteurs pour définir et élaborer les stratégies et démarches de développement.

2. Documents applicables et de référence

Sujet du projet (cf « projet1-2-3.pdf »).
Document de spécification : PBV_STB_1.0

3. Terminologie et sigles utilisés

- Module noyau : Programme qui peut être chargé dynamiquement dans le noyau.
- SSD : Solid State Disk, périphérique de stockage qui utilise de la mémoire flash. Les performances sont élevées et les consommations d'énergie basses mais le coût par Go est élevé.
- HDD : Hard Drive Disk, périphérique de stockage de masse. Il consomme plus et ses performances sont moindres qu'un SSD mais le prix d'acquisition est beaucoup moins élevé.
- Protocole : Programme qui gère les communications entre plusieurs entités.
- PBV : Périphérique Bloc Virtuel. Il sera constitué de 0 ou 1 SSD, et d'au moins 1 HDD. Classiquement, il sera composé d'un périphérique de chaque type.
- SGF : Système de Gestion de Fichiers, aussi appelé gestionnaire de système de fichiers ici. Programme gérant les opérations effectuées sur un périphérique de stockage.
- Pilote : Programme gérant les opérations entre un périphérique et le reste du système.
- Mémoire cache : Mémoire non volatile d'accès rapide. Les données ne sont pas liées à un processus en cours d'exécution.

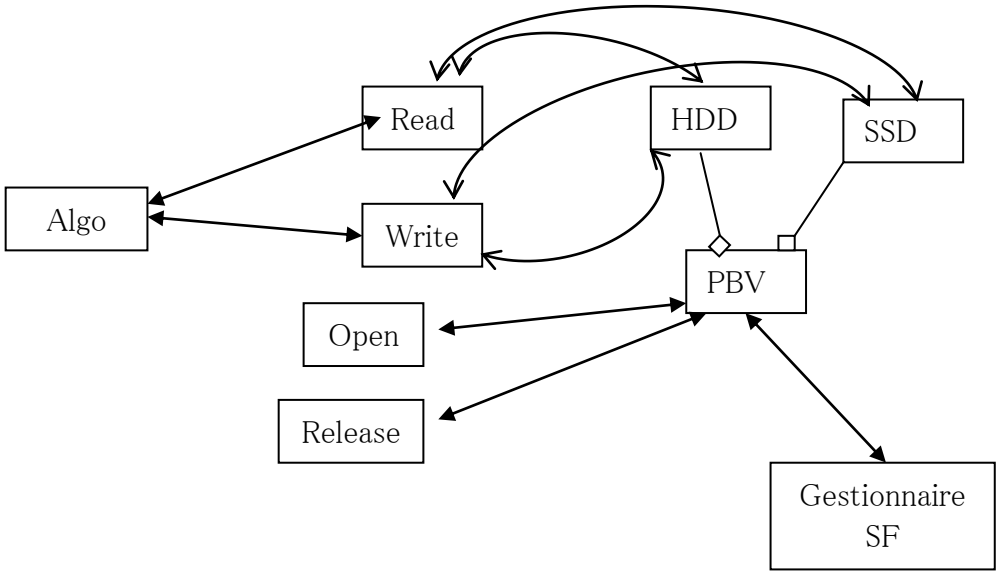
4. Architecture physique du matériel utilisé

- Le but de ce projet étant d'implanter une optimisation de fonctionnement de SSD/HDD simultanée, tout processeur, même de faible puissance, est suffisant.
- Périphériques et matériels spécifiques : par la nature du projet, un SSD et un HDD sont requis.
- Système d'exploitation : le logiciel devra être opérationnel sur un OS de type Linux.
- Liste des produits logiciels nécessaires : seront nécessaires un OS de type Linux, des contrôleurs SATA et/ou IDE.

5. Architecture statique du logiciel

1. Structure logique

Le schéma ci-dessous représente la structure logique de notre solution proposée. Un second schéma, plus global et plus représentatif de la problématique, est disponible en annexe.



2. Description du constituant « Open »

-Alloue les ressources nécessaires à la manipulation du périphérique bloc virtuel. Incrémente le compteur d'utilisateurs.

- Retourne le résultat de l'ouverture du PBV (succès/échec).
- Développé en C/C++
-Taille à définir. Complexité faible.

3. Description du constituant « Release »

- « Nettoie » les données utilisées par les opérations d'écriture/lecture. Décrémente le compteur d'utlisateurs. Retourne un entier, dont la valeur diffère selon la réussite de l'action.
- Développé en C/C++
- Taille à définir. Complexité faible.

4. **Description du constituant «Read»**

- Recherche de la donnée à lire. Retourne les blocs lus.
- Dépend des modules Algo
- Développé en C/C++
- Méthode Agile
- Taille et complexité à définir.

5. **Description du constituant « Write »**

- Demande d'allocation de blocs auprès des périphériques de stockage. Ecrit les données sur le disque, après appel du module Algo. Retourne le nombre de bits réellement écrits.
- Dépend des modules Algo
- Développé en C/C++
- Méthode Agile
- Taille et complexité à définir

6. **Description du constituant « Algo »**

- Ensemble de sous-composants permettant la gestion du cache et la synchronisation. Il intervient lorsqu'une opération de type Read ou Write est effectuée. Par exemple, il permettra de savoir si une donnée doit être écrite sur le SSD plutôt que sur le HDD.
- Dépend des modules Write et Read.
- Développé en C/C++
- Méthode Agile
- Taille : élevée. Complexité : Faible. Complication : Elevée

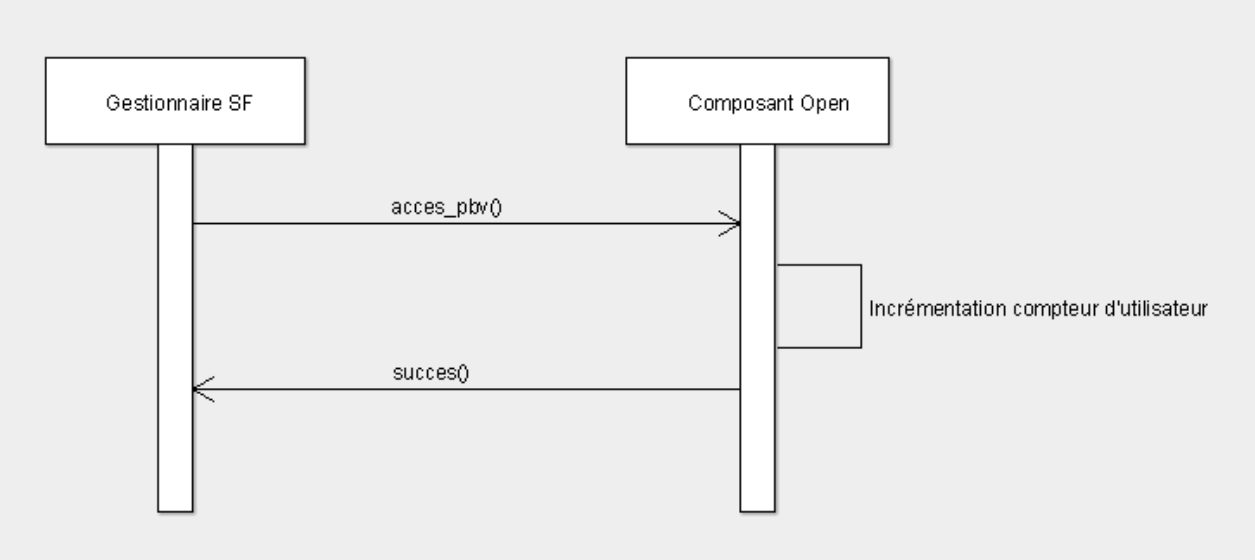
6. Justifications techniques

Langage : Le langage choisi est le C, car celui-ci est un des langages spécifiques à la programmation système. Il permettra ainsi de travailler à un niveau relativement bas et nous permettra par exemple de développer des modules noyau. C'est aussi dans ce langage que nous sommes le plus à l'aise.

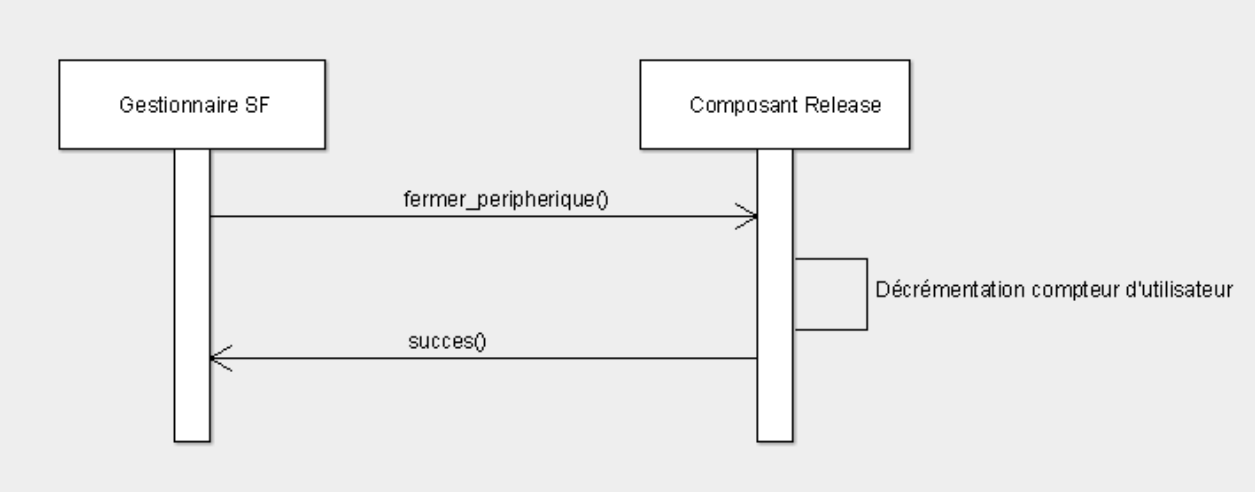
Périphérique Bloc Virtuel : Cette approche, par opposition à la ré-implémentation d'un système de fichiers, nous permet une plus grande flexibilité. Ainsi, la simulation d'un périphérique bloc virtuel offre la possibilité de ne pas reposer sur le type de système de fichiers et garantit un meilleur portage.

7. Fonctionnement dynamique

- Ouverture du périphérique (flot principal) :
  Composants mis en jeu : le gestionnaire de système de fichiers, Open.
  Le PBV a auparavant bien été enregistré.



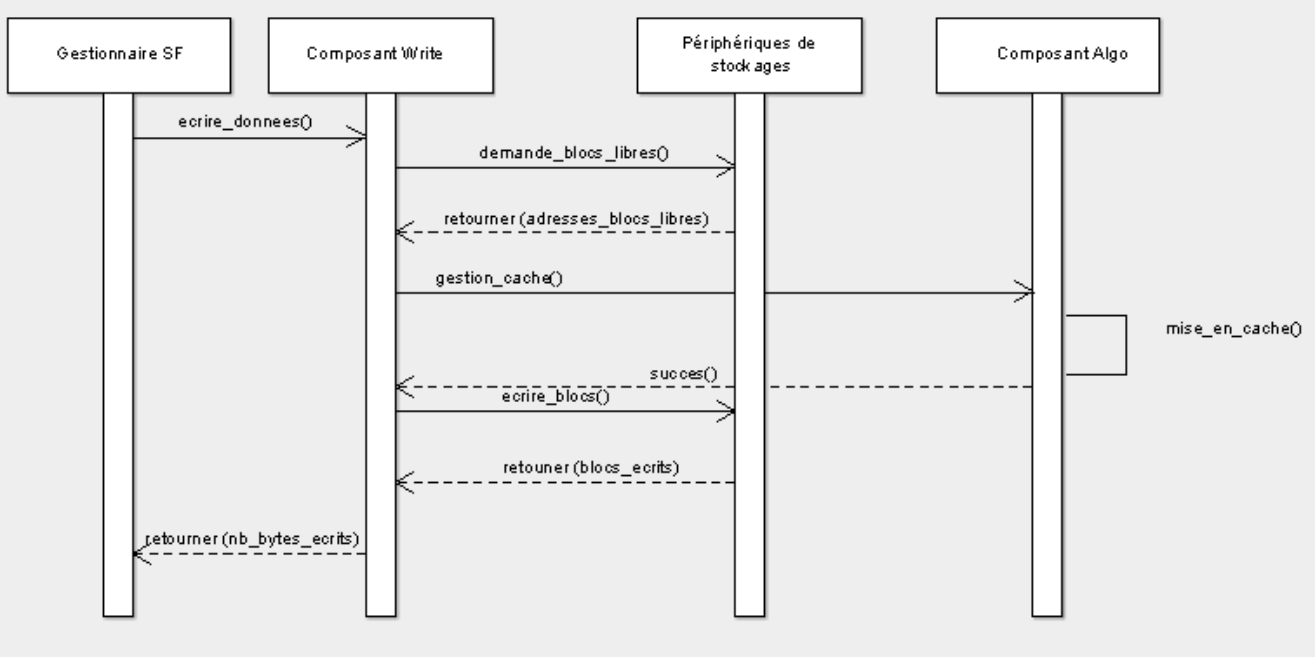
- Fermeture du périphérique (flot principal) :
  Composants mis en jeu : le gestionnaire de système de fichiers, Release.
  Le PBV a auparavant bien été enregistré.



- Ecriture de blocs de données (flot principal) :

Composants mis en jeu : Write, Algo, le gestionnaire de système de fichiers et les périphériques de stockage.

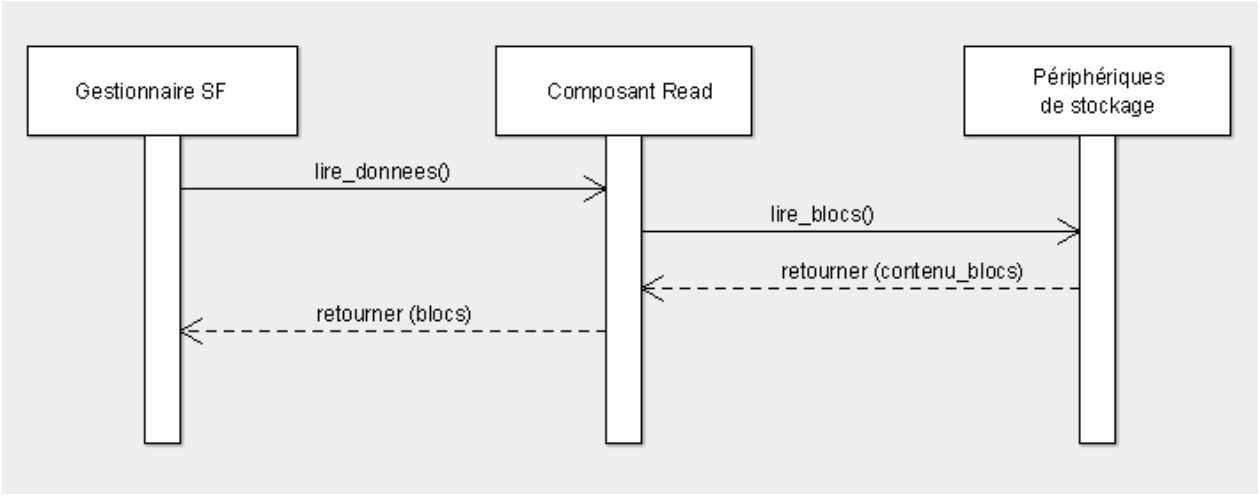
Le gestionnaire de système de fichiers a déjà autorisé l'ouverture du fichier en écriture.



- Lecture de blocs de données (flot principal) :

Composants mis en jeu : Read, le gestionnaire de système de fichiers et les périphériques de stockage.

Le gestionnaire de système de fichiers a déjà autorisé l'ouverture du fichier en lecture.



8. Traçabilité

Se reporter à la spécification technique de besoin (STB) pour associer les numéros de fonctions à leur description.

Exigences		F-FR-20	F-FO-20	F-FO-30	F-FO-40	F-FQ-10	F-FQ-20
Composants	*Open*	*x*	*x*				
	Release	*x*	*x*				
	Read	*x*	*x*				
	Write	*x*	*x*				
	Algo			*x*	*x*	*x*	*x*

9. Annexe

Schéma de fonctionnement global du pilote et des intéractions avec les périphériques et le noyau.