

## **SPECIFICATION TECHNIQUE DE BESOIN**

**Version :** 0.2

**Date :** 03/11/2016

**Rédigé par :** L'équipe SmartLogger

**Relu par :** L'équipe SmartLogger

**Approuvé par :** ---

*Signature du superviseur :*

**Objectif :** Ce document est destiné à traduire les besoins des utilisateurs du logiciel et à établir une référence pour sa validation. Il doit être élaboré en coopération avec le demandeur puis approuvé par ce dernier. Son but est de recenser les principales exigences que l'équipe de développement s'engage à satisfaire dans le cadre du projet.



## **HISTORIQUE DE LA DOCUMENTATION**

<b>Version</b>	<b>Date</b>	<b>Modifications réalisées</b>
0.1	03/11/16	Création
0.2	10/11/16	Augmentation et correctifs

## **1. Objet :**

Le but du projet est de créer un système, permettant d'alerter l'utilisateur sur des données en provenance d'applicatifs défectueux dans l'optique de faciliter leurs correctifs. Une fois produit, ce système sera utilisé par l'entreprise cliente à leurs propres fins.

L'objectif est donc de mettre en place un système d'alarme qui disposerait de ses propres capacités d'apprentissage. Sa tâche principale sera de prioriser les différentes données à traiter selon leur niveau de criticité et d'adopter une stratégie de prévention adaptée.

Sur le long terme, ce dernier devra améliorer son analyse en exploitant ses résultats précédents afin de constamment améliorer le taux de confiance des alertes émises. L'enjeu sera donc de créer un système combinant facultés d'analyse et de prédiction.

Dans cette optique, l'entreprise client nous a recommandé l'emploi de certaines technologies qui, pour la plupart, sont connues et utilisées pour leurs propres projets.

### Ceci inclut :

- Spring Boot : afin de réaliser la partie applicative.
- Angular 2 : pour la réalisation de l'interface graphique.
- Spark : pour la mise en place des algorithmes.

Le développement du projet entier ne repose sur aucune fonctionnalité ou logiciel existant.

De ce fait, l'équipe projet devra concevoir le fonctionnement du système dans son intégralité et en assurer la production.

Cependant, comme demandé par le client, le produit final devra respecter les contraintes suivantes :

- Il devra analyser des flux de données de type Shinken, Logstash et Apache Kafka.
- Il alertera les opérateurs en employant des moyens de communication utilisés par l'entreprise, tels que l'application Slack, l'envoi de mails ou par notification SMS.
- Il pourra s'adapter à des flux de données ou méthodes d'alerte non prédéfinies qui seront implantées dans le système par de futurs utilisateurs.
- Enfin, il devra fonctionner sur de longues périodes de fonctionnement et, dans l'idéal, être opérationnel indéfiniment.

## **2. Documents applicables et de référence**

- Le document de spécification client : SmartLogger.pdf

### **3. Terminologie et sigles utilisés**

#### **Définitions et Notions :**

**Log** : Un fichier log ou «log» est un fichier contenant l'historique des événements d'un processus en particulier. Ici, nous considérerons des logs issus d'applications WEB externes.

**Événement** : Un événement représente un changement au sein d'un processus. Il peut s'agir d'un changement en mémoire (ajout d'une donnée), un changement dans l'interface (clic de souris), etc.

**Flux de données** : Un flux de données représente une quantité potentiellement illimitée de données qui est manipulée sur un laps de temps défini afin d'en extraire une certaine quantité.

**Traitement en temps réel** : Un traitement est réalisé en temps réel, si le système qui l'effectue peut adapter sa vitesse à l'évolution de ce dernier.

**Machine Learning (ML)** : Champ d'étude de l'intelligence artificielle ayant pour but de faire évoluer, au cours du temps, la façon dont un système effectue un même traitement. Par extension, un algorithme de Machine Learning est un algorithme permettant d'implanter une telle capacité d'apprentissage à un système.

**User Interface (UI)** : Désigne une application pouvant être manipulée par un utilisateur dans le but d'utiliser un système.

#### **Technologies employées :**

**SGBD** : Système de gestion de base de données, permet de stocker, manipuler et organiser un (très) grand nombre de données diverses.

**Shinken** : Application permettant la surveillance de systèmes et de réseaux. Elle surveille les hôtes et services spécifiés, lançant une alerte lorsque les systèmes vont mal et quand ils vont mieux.

**Logstash** : Outil informatique permettant de gérer des événements et des logs.

**Apache Kafka** : Projet open-source visant à fournir un système unifié en temps réel à latence faible pour la manipulation de flux de données.

**Slack** : Logiciel de gestion de projets, principalement utilisé pour son système de communication entre membres d'équipes de projet.

**MongoDB** : SGBD orienté documents, il permet de manipuler des données sans avoir à concevoir la façon dont ces dernières seront gérées en interne.

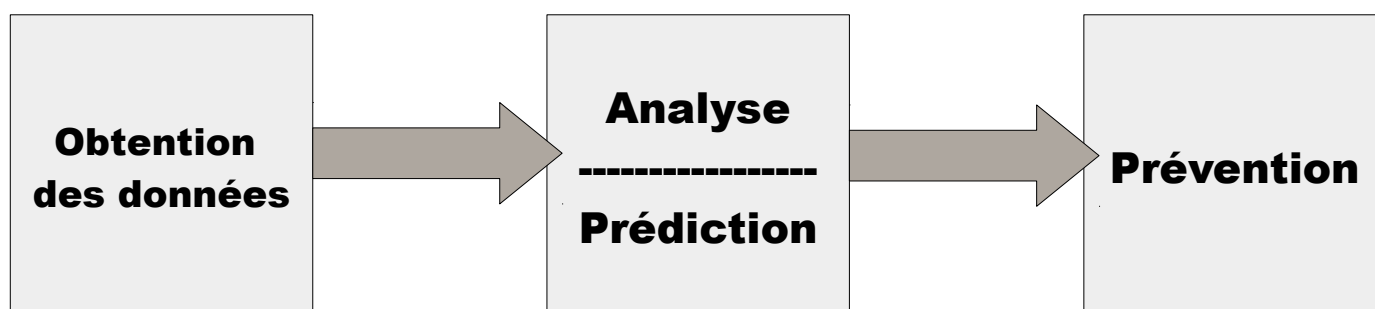
**NoSQL** : Désigne une certaine famille de SGBDs pouvant manipuler de plus grands volumes de données en outrepassant d'anciennes règles pré-établies sur les autres types de SGBD.

## 4. Exigences fonctionnelles

### 4.1. Présentation de la mission du produit logiciel

Le logiciel final, bien que scindé en plusieurs parties distinctes, sera déployé sur un serveur externe appartenant au client puisqu'il nécessite divers flux de données issus de systèmes extérieurs. Il agira en tant qu'intermédiaire entre les logs et les opérateurs de maintenance, prenant ainsi position dans ce type de procédure.

Sa fonction principale reposera sur la mécanique suivante :



- La première phase consiste en la récupération des données. En effet, vu qu'elles proviennent de sources extérieures au système, il faudra veiller à leur conformité mais également à la cohérence de leur contenu. Le but étant de protéger la machine contre des attaques externes (e.g. : injections de code) ou échantillons de données qui tronqueraient le modèle de prévision mis en place.
- La deuxième phase consiste en l'analyse de ces dites données. Celle-ci s'opère à deux niveaux :
  - Une analyse brute qui déterminera l'état de l'applicatif en fonction des données physiques présentes sur le log. Il permet de déterminer **l'état d'une application à un instant donné**.
  - Une analyse prédictive qui emploiera des algorithmes de Machine Learning pour déterminer **l'état d'une application dans un futur proche**.Cette analyse permettra d'identifier des défaillances dans certains logiciels. Ces défaillances seront alors classifiées puis triées selon leur niveau de criticité.
- La troisième phase exploite la classification précédente : si le résultat d'une analyse témoigne d'un état critique, le système va alors alerter les différents opérateurs concernés au moyen de canaux de communication (Slack, Mail, SMS).

Néanmoins, pour fournir une telle capacité de prédiction au système, il va falloir fournir des jeux d'essais et offrir la possibilité d'ajuster les traitements effectués afin de perfectionner le processus d'analyse prédictive.

Le système devra donc proposer à un utilisateur, via une interface graphique, de pouvoir modifier la façon dont un échantillon de données précis doit être convenablement traité, mais également de pouvoir rajouter des échantillons supplémentaires pour augmenter la capacité prédictive du logiciel.

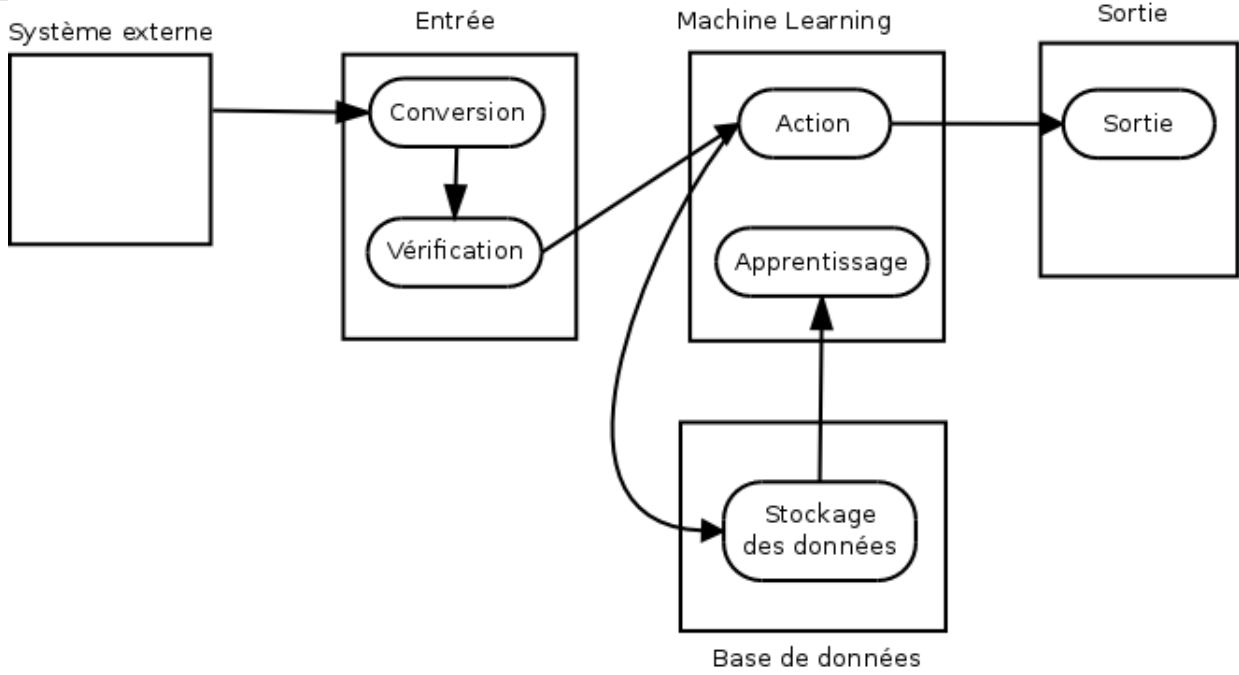
Ainsi, en fonction de ses tâches, le système accordera un certain niveau de permission selon le rôle à assurer, ce qui permet de partitionner la fonction de chaque utilisateur vis-à-vis du système.

De ce fait, le système distinguera 4 profils d'utilisateurs différents :

- Les **systèmes informatiques** : les principaux bénéficiaires du système.
- Les **entraîneurs** : Des utilisateurs qui réalisent l'apprentissage initial du système en fournissant des jeux d'essais.
- Les **testeurs** : Des entraîneurs particuliers pouvant modifier le comportement de l'analyse.
- Les **opérateurs** : Des utilisateurs polyvalents pouvant assurer le rôle de testeur ou donner des ordres directs au système afin de procéder à la réutilisation de données exploitables.

**On décrira alors les différents modes de fonctionnement en 4 cas distincts :**

4.2. Mode de fonctionnement principal

Fonctionnement principal	
<b>Acteurs concernés</b>	Le système (Entrée, ML, DB, Sortie), un système externe client.
<b>Description</b>	<p>Cas d'utilisation de la machine lors de son mode de fonctionnement principal.</p> <p>La machine reçoit des données d'un système externe qui sont converties, vérifiées, puis stockées dans la base de données. Elle effectuera alors son analyse et utilisera les résultats pour affiner son comportement.</p> <p>Si une alerte critique est détectée, elle procédera à l'alerte des opérateurs associé à l'appliquatif défaillant.</p>
<b>Pré-conditions</b>	a) Le système est opérationnel. b) Le système dispose de données à traiter. c) Le système est inactif (ne réalise aucune autre tâche).
<b>Événement déclenchant</b>	Réception d'un volume de données quelconque provenant d'un système externe.
<b>Conditions d'arrêt</b>	a) Le système termine son cycle d'exécution. b) Une alerte a été lancée, suite au traitement des données.
<b>Description du flot d'événements principal :</b>	
 <pre> graph LR     SE[Système externe] --&gt; C[Conversion]     subgraph Entrée         C --&gt; V[Vérification]     end     V --&gt; A[Action]     subgraph ML [Machine Learning]         A --&gt; S[Sortie]         A --&gt; SD[Stockage des données]         SD --&gt; AP[Apprentissage]         AP --&gt; A     end     subgraph BD [Base de données]         SD     end           </pre>	

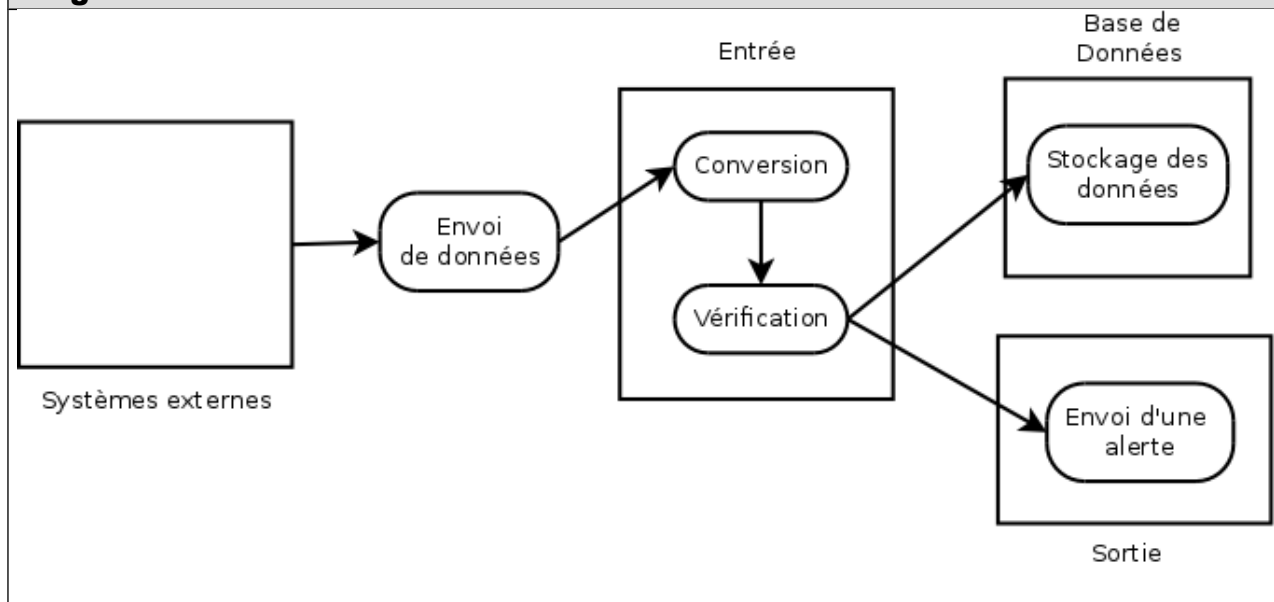
Ce mode de fonctionnement comporte un sous-cas de fonctionnement selon la cohérence des données reçues :



#### 4.2.1. Fonctionnement lors du stockage de données erronées

Stockage de données erronées	
<b>Acteurs concernés</b>	Le système (Entrée, DB, Sortie), un système externe client.
<b>Description</b>	<p>Intervient lorsqu'un système externe envoie des données erronées ou incohérentes au système.</p> <p>Si une erreur apparaît lors de la conversion ou de la vérification des données, on procède à leur stockage en attendant leur possible exploitation future (format inconnu, etc.)</p> <p>On envoie également une alerte afin de notifier les opérateurs de cette anomalie.</p>
<b>Pré-conditions</b>	Idem 4.1
<b>Événements déclenchants</b>	<p>a) La conversion des données a échoué.</p> <p>b) La vérification de la validité des données a échoué.</p>
<b>Conditions d'arrêt</b>	<p>a) Les données ont été stockées dans la base de données.</p> <p>b) Une notification à un/des opérateur(s) a été envoyée.</p>

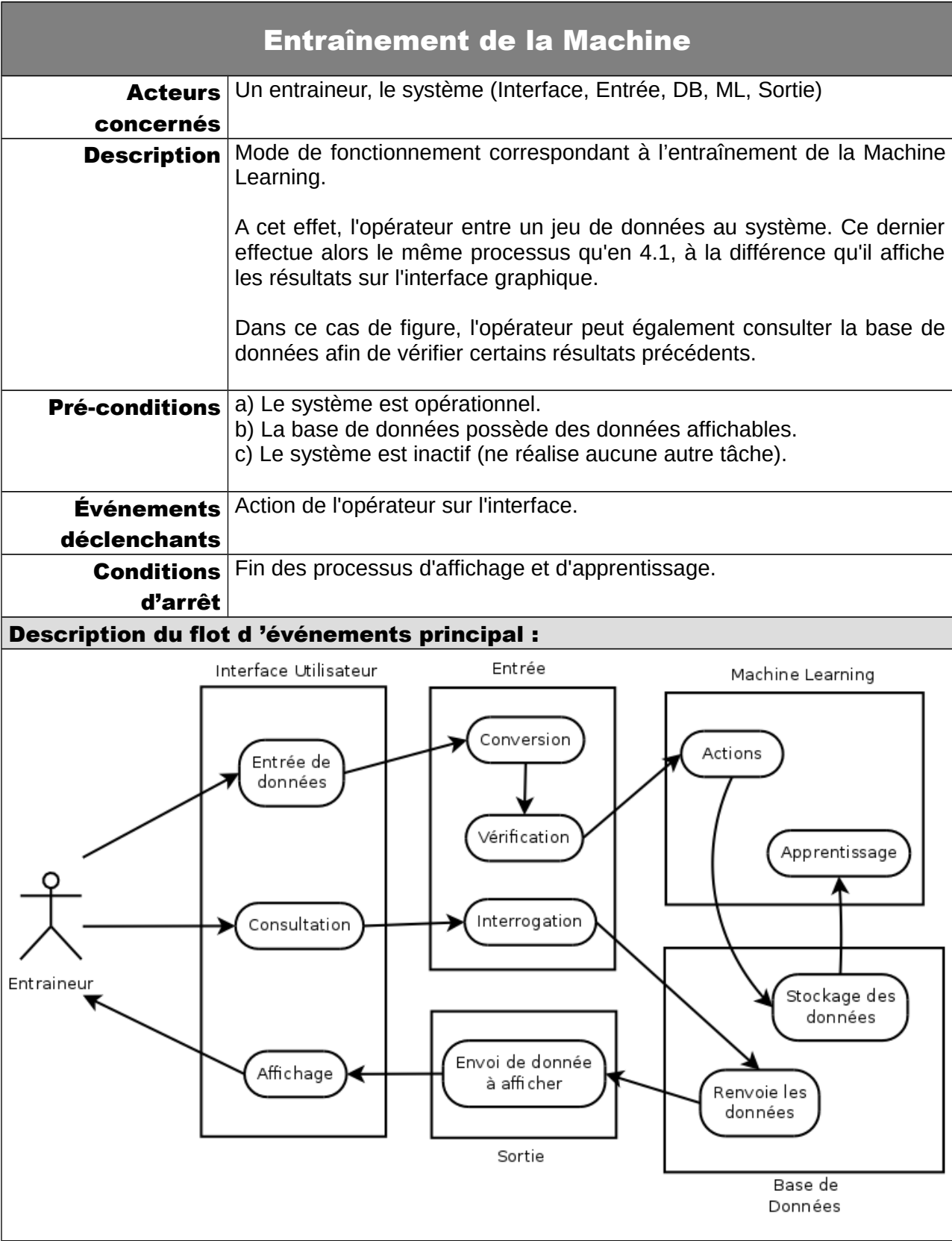
### Diagramme de cas d'utilisation :



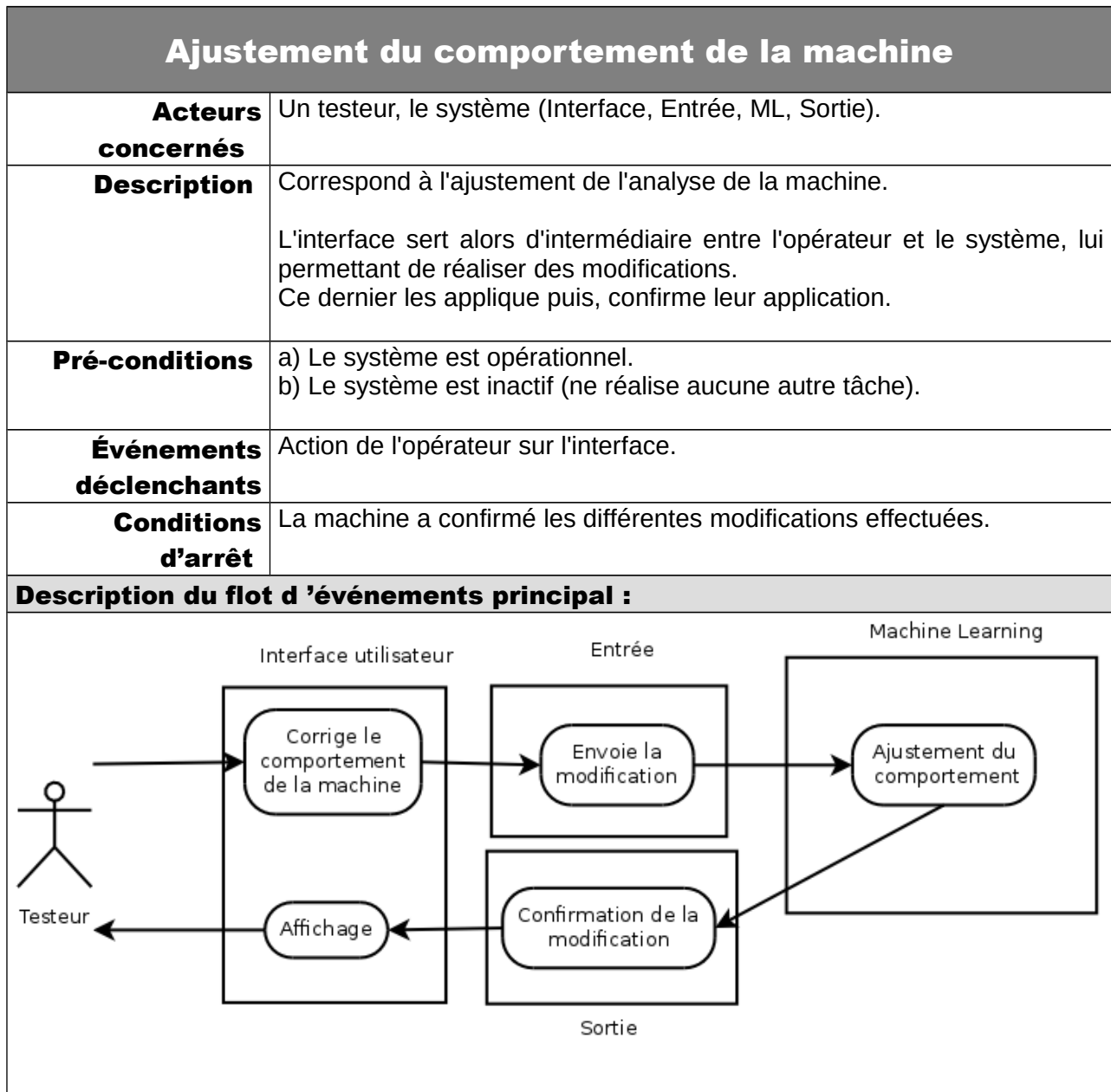
4.3. Réutilisation de données

Réutilisation de données	
Acteurs concernés	Un opérateur, le système (Interface, Entrée, DB, ML, Sortie)
Description	<p>Cas d'utilisation correspondant à un ordre direct d'un opérateur. Intervient lorsqu'on souhaite effectuer des traitements sur des données auparavant incorrectes ou pour réévaluer le comportement du système.</p> <p>Le système va effectuer un traitement sur l'échantillon de données désigné par l'opérateur. Le fonctionnement est alors identique à 4.1.</p>
Pré-conditions	<p>a) Le système est opérationnel.</p> <p>b) Le système dispose de données à traiter.</p> <p>c) Le système est inactif (ne réalise aucune autre tâche).</p>
Événements déclenchants	<p>a) Envoi de force de données invalides.</p> <p>b) La conversion des données a échoué.</p> <p>c) La vérification de la validité des données a échoué.</p>
Conditions d'arrêt	<p>a) Les données sont stockées dans la base de données.</p> <p>b) Une alerte est envoyée.</p>
Description du flot d 'événements principal :	
<pre>graph LR     subgraph Interface_utilisateur [Interface utilisateur]         A((Force un envoi de données))     end     subgraph Entree [Entrée]         B((Conversion))         C((Vérification))     end     subgraph Base_de_Donnees [Base de Données]         D((Récupération des données))         E((Stockage des données))     end     subgraph Machine_Learning [Machine Learning]         F((Actions))     end     subgraph Sortie [Sortie]         G((Envoi d'une Alerte))     end      A --&gt; B     B --&gt; C     C --&gt; D     D --&gt; F     F --&gt; E     F --&gt; G</pre> <p>The diagram illustrates the main event flow across several system components. It begins with an 'Opérateur' (Operator) who triggers the 'Force un envoi de données' (Force data sending) event in the 'Interface utilisateur' (User Interface). This event leads to the 'Conversion' event in the 'Entrée' (Input) component, which then flows to the 'Vérification' (Verification) event. From 'Vérification', the flow goes to the 'Récupération des données' (Data retrieval) event in the 'Base de Données' (Data Base) component. This leads to the 'Actions' event in the 'Machine Learning' component. From 'Actions', the flow branches: one path goes to the 'Stockage des données' (Data storage) event in the 'Base de Données', and another path goes to the 'Envoi d'une Alerte' (Alert sending) event in the 'Sortie' (Output) component.</p>	

4.4. Entraînement du système



#### 4.5. Ajustement du comportement de la machine



## 5. Exigences opérationnelles

P	Référence	Désignation
	<b>OP-1</b>	<b>Le système doit fonctionner sans interruption</b>
	<b>OP-1.1</b>	La maintenance du système doit s'opérer le plus rapidement possible
	<b>OP-1.2</b>	Le nombre d'arrêts du système doit être le plus faible possible, sur toute la période de sa durée de vie. (Hors opérations d'ajout de modules externes)
	<b>OP-1.3</b>	Le système doit être conçu pour fonctionner sur un serveur de type Linux.
	<b>OP-2</b>	<b>Le traitement des résultats doit être adapté à la criticité détectée</b>
	<b>OP-2.1</b>	Les détections de failles critiques doivent mener à une levée d'alerte
	<b>OP-2.2</b>	Les détections de failles mineures doivent être signalées avec une importance moindre qu'une alerte.
	<b>OP-2.3</b>	Les résultats sans importance ne doivent pas être signalés
	<b>OP-3</b>	<b>Toute donnée manipulée par le système doit être enregistrée</b>
	<b>OP-3.1</b>	Les données incohérentes ou non vérifiées seront stockées sans traitement préalable
	<b>OP-3.2</b>	Les résultats issus d'une analyse seront stockés avec les données brutes correspondantes
	<b>OP-3.3</b>	Un utilisateur du système doit pouvoir manipuler d'anciennes données déjà utilisées
	<b>OP-4</b>	<b>Le système doit fonctionner de manière cyclique</b>
	<b>OP-4.1</b>	Les résultats des analyses seront envoyés à intervalle régulier
	<b>+ OP-4.1.1</b>	Un script devra assurer l'envoi de résultats toutes les X secondes
	<b>+ OP-4.1.2</b>	Même si une analyse est en cours, le système doit procéder à un envoi de l'ensemble des résultats obtenus
	<b>OP-4.2</b>	Chaque cycle d'exécution doit commencer par un traitement des données reçues
	<b>OP-4.3</b>	Chaque cycle d'exécution doit se terminer par un apprentissage des données traitées
	<b>OP-5</b>	<b>Le système doit prévenir toute tentative d'utilisation incorrecte des données</b>
	<b>OP-5.1</b>	La consultation des données du système ne peut s'effectuer qu'au travers de l'interface
	<b>OP-5.2</b>	Les données traitées doivent subir un chiffrement avant leur stockage dans la base de données
	<b>OP-5.3</b>	Le système doit vérifier l'intégrité des données lues en entrée, avant de procéder à toute autre phase de traitement

Echelle de priorité : **Indispensable**, **Important**, **Optionnel**

## 6. Exigences d'interface

P	Référence	Désignation
	<b>IN-1</b>	<b>L'interface d'entrée du système doit pouvoir traiter tout type de log</b>
	<b>IN-1.1</b>	L'interface d'entrée doit offrir la possibilité d'utiliser plusieurs types de données prédéfinis
	<b>+ IN-1.1.1</b>	Le système doit pouvoir utiliser des données de type Apache Kafka
	<b>+ IN-1.1.2</b>	Le système doit savoir manipuler des données de type Shinken
	<b>+ IN-1.1.3</b>	Le système doit pouvoir utiliser des données logstash
	<b>IN-1.2</b>	L'interface d'entrée doit pouvoir être modifiée afin d'accepter de nouveaux types de données
	<b>IN-2</b>	<b>L'interface de sortie doit disposer de plusieurs fonctionnalités d'alerte et de notification</b>
	<b>IN-2.1</b>	L'interface de sortie doit disposer de plusieurs fonctionnalités prédéfinies afin d'effectuer ses alertes
	<b>+ IN-2.1.1</b>	L'alerte doit pouvoir être donnée via l'application Slack
	<b>+ IN-2.1.2</b>	L'alerte doit pouvoir être donnée par Mail
	<b>+ IN-2.1.3</b>	L'alerte peut être donnée par SMS
	<b>IN-2.2</b>	L'interface de sortie doit pouvoir s'adapter à des formats de sorties supplémentaires

Echelle de priorité : **Indispensable**, **Important**, **Optionnel**

## 7. Exigences de qualité logicielle et de réalisation

P	Référence	Désignation
	<b>QR-1</b>	<b>La compréhension du code source doit être facilitée</b>
	<b>QR-1.1</b>	Le code source fourni doit être propre et commenté
	<b>QR-1.2</b>	Les identificateurs de variables et de fonctions doivent être significatifs de leur utilité
	<b>QR-1.3</b>	Les identificateurs utilisés doivent être issus de la langue anglaise
	<b>QR-1.4</b>	Le code source final devra être correctement documenté
	<b>QR-2</b>	<b>Le type de programmation employé doit être le plus modulaire possible</b>
	<b>QR-2.1</b>	Les nouveaux types de flux d'entrées doivent être facilement implémentables, au moyen de nouveaux modules d'entrée
	<b>QR-2.2</b>	L'ajout de nouvelles méthodes d'alerte ou de notification souhaitées, doit pouvoir s'effectuer de façon simple, en ajoutant de nouveaux modules de sortie.
	<b>QR-2.3</b>	Les modules algorithmiques gérant la partie d'analyse prédictive du système doivent être interchangeables.

Echelle de priorité : **Indispensable**, **Important**, **Optionnel**

