**Analyse et Conception – Etude des contrôles dans une application**

Table des matières

[1 Objectif 2](#_Toc450892721)

[2 Analyse du concept Contrôle 3](#_Toc450892722)

[2.1 Dégrossir le problème (Collecter les besoins, exigences et règles de gestion) 3](#_Toc450892723)

[2.1.1 Collecter les besoins, exigences et règles de gestion (RG) liés aux contrôles auprès de la MOA 3](#_Toc450892724)

[2.1.2 Appliquer les règles de l’art 4](#_Toc450892725)

[2.1.3 Identifier quelques types généraux de contrôles 4](#_Toc450892726)

[2.1.4 Identifier les acteurs et imaginer comment l'application devra rendre compte des contrôles (rapports de contrôle, logs) 10](#_Toc450892727)

[2.2 Identifier les concepts 11](#_Toc450892728)

[2.2.1 Identifier les concepts nécessaires à minima pour l'acteur le plus exigeant (Quand-Qui-Quoi-Qu'est ce - Que-Quel-Où) 11](#_Toc450892729)

[2.2.2 Identifier les concepts liés au cycle de vie d’une application 13](#_Toc450892730)

[2.2.3 Simuler le déclenchement des contrôles par un fournisseur de données 15](#_Toc450892731)

[2.2.4 Imaginer la gestion des contrôles par l'Administrateur 16](#_Toc450892732)

[2.2.5 Imaginer comment les acteurs utiliseront les rapports de contrôle (tris, requêtes, …) 17](#_Toc450892733)

[2.2.6 Lister les questions types que se posera la MOA 17](#_Toc450892734)

[2.2.7 Capturer les concepts 17](#_Toc450892735)

[2.3 Synthétiser 17](#_Toc450892736)

[2.3.1 Synthèse des acteurs retenus 17](#_Toc450892737)

[2.3.2 Synthèse des exigences retenues 17](#_Toc450892738)

[2.3.3 Synthèse du modèle conceptuel retenu 17](#_Toc450892739)

[2.4 Affiner et Structurer les concepts 17](#_Toc450892740)

[2.5 Etudier l'enchaînement des contrôles 18](#_Toc450892741)

[2.6 Etudier la réutilisation des contrôles 18](#_Toc450892742)

[3 Conception des contrôles 18](#_Toc450892743)

[3.1 "jouer" des contrôles 18](#_Toc450892744)

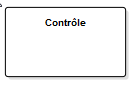
[4 Administration des contrôles dans l'application 18](#_Toc450892745)

[5 Capitalisation de la méthode 18](#_Toc450892746)

# Objectif

Modéliser les concepts utilisés par les contrôles dans une application informatique afin d'élaborer des classes de contrôle.

On cherche ici à définir ce que doit "**savoir**" et "**savoir faire**" un **contrôle** dans une application informatique.



# Analyse du concept Contrôle

## Dégrossir le problème (Collecter les besoins, exigences et règles de gestion)

### Collecter les besoins, exigences et règles de gestion (RG) liés aux contrôles auprès de la MOA

L'objectif est de comprendre les premiers besoins et exigences de la **M**aîtrise d'**O**uvr**A**ge (MOA) en matière de contrôle.

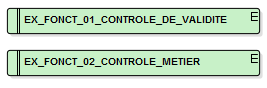
Toute application informatique a besoin de **contrôler** la validité des ressources qu'elle accepte en entrée ainsi que la qualité de ses livrables. La **M**aîtrise d'**O**uvr**A**ge (MOA) a donc une *exigence fonctionnelle de contrôle de validité* des fichiers entrants.

Par exemple, une application informatique qui traite des fichiers devra vérifier que les fichiers entrants sont non-null, existants, pas des répertoires, non vides, conformes au format attendu, …

De même, la MOA a toujours une *exigence fonctionnelle de contrôle métier* des données. Il s'agit de s'assurer que les données métier gérées par l'application satisfont des **règles de gestion (RG)** définies par la MOA.

#### Concepts et Exigences mis en évidence





### Appliquer les règles de l’art

Certains besoins et exigences ne sont pas directement exprimés par la MOA lors des interviews mais ils s’imposent d’eux-mêmes à l’informaticien : les **règles de l’art**.

L’informaticien doit par exemple toujours s’assurer que n’importe quel type de fichier ne peut être entré dans l’application même si la MOA n’a pas spécifié ce point lors des interviews.

La MOA a donc toujours une *exigence fonctionnelle* implicite *d’application des règles de l’art*.

#### Concepts et exigences mis en évidence

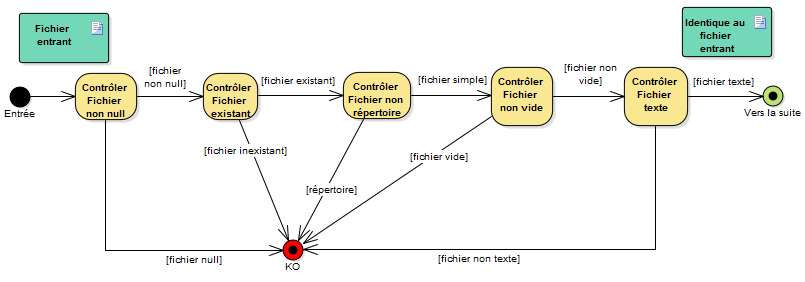


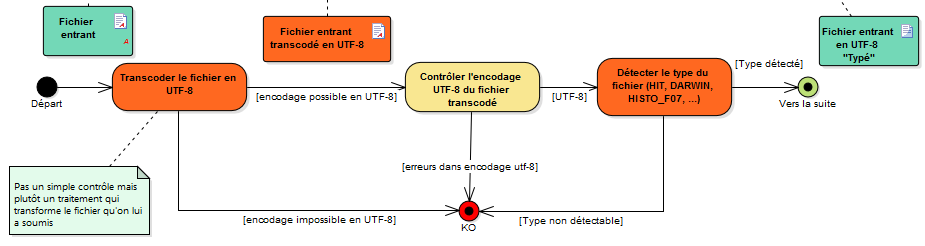


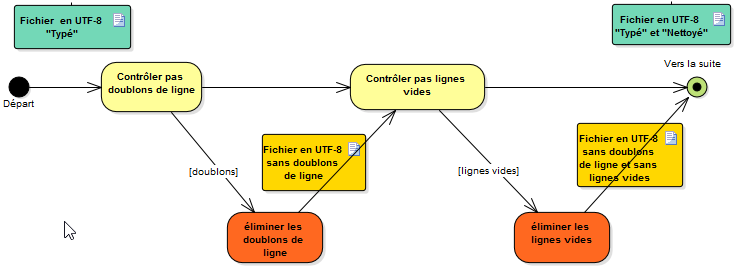
### Identifier quelques types généraux de contrôles

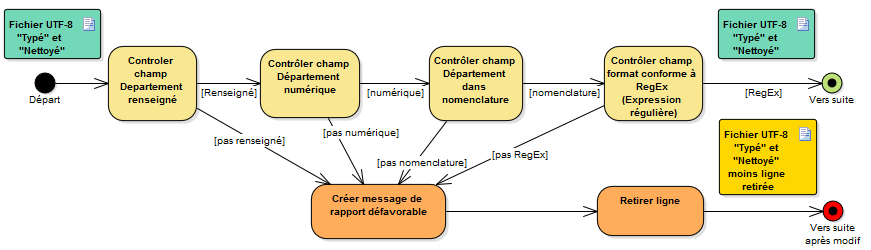
**Détecter quelques contrôles** et traitements que l'application devra obligatoirement exécuter. L'idée est de dégrossir le problème et de se faire une idée de ce qu'est le concept de contrôle. On applique notamment des **règles de l’art** (comme par exemple, il est toujours inutile d’accepter en entrée d’une application informatique un fichier vide).

Par exemple, en analysant le début du traitement d'un fichier dans l'application TraficWeb :









#### Résumé des contrôles sous forme tabulaire

| **ordre** | **Lieu** | **Niveau**  **(Portée)** | **Type** | **META-NOM du contrôle** |
| --- | --- | --- | --- | --- |
| 1 | Surface | Fichier | Nullité | Contrôle de surface de non-nullité d'un fichier |
| 2 | Surface | Fichier | Existence de fichier | Contrôle de surface d'existence d'un fichier |
| 3 | Surface | Fichier | Non-répertoire | Contrôle de surface de simplicité (non-répertoire) d'un fichier |
| 4 | Surface | Fichier | Contenu (non vide) | Contrôle de surface de contenu (non vide) d'un fichier |
| 5 | Surface | Fichier | Textuel | Contrôle de surface du caractère textuel d'un fichier |
| 6 | Surface | Fichier | Transcodage\_UTF8 | Transcodage de surface du fichier en UTF-8 |
| 7 | Surface | Fichier | Encodage\_UTF8 | Contrôle de surface de l'encodage en UTF-8 du fichier |
| 8 | Surface | Fichier | Détection\_Type\_HIT | Détection de surface du type HIT d'un fichier |
| 9 | Surface | Fichier | Détection\_Type\_HISTO\_F07 | Détection de surface du type HISTO\_F07 d'un fichier |
| 10 | Surface | Fichier | Détection\_Type\_HISTO\_F08 | Détection de surface du type HISTO\_F08 d'un fichier |
| 11 | Surface | Fichier | Détection\_Type\_DARWIN | Détection de surface du type DARWIN csv d'un fichier |
| 12 | Surface | Fichier | Détection\_Type\_FEOR\_XML | Détection de surface du type FEOR XML d'un fichier |
| 13 | Surface | Inter-Lignes | Détection de doublons de lignes | Détection-Elimination de surface des doublons de lignes |
| 14 | Surface | Ligne | Nullité | Contrôle de surface de nullité d'une ligne |
| 15 | Surface | Ligne | Renseigné | Contrôle de surface de renseignement d'une ligne (non null ou vide) |
| 16 | Surface | Champ | Renseigné | Contrôle de surface de renseignement d'un champ (non null ou vide) |
| 17 | Surface | Champ | Format Numérique | Contrôle de surface de format numérique d'un champ |
| 18 | Surface | Champ | Dans Nomenclature | Contrôle de surface de nomenclature d'un champ |
| 19 | Surface | Champ | Regex (expression régulière) | Contrôle de surface d'expression régulière d'un champ |

#### Concepts, exigences et RG mis en évidence

L'examen des diagrammes d'activité des contrôles précédents permet de dégager de nouveaux concepts, exigences et règles de gestion.

Par exemple, lorsqu'un fichier entrant est inexistant ou vide, l'application doit s'arrêter. On voit donc apparaître le concept de **Suite à donner** à un contrôle (fichier refusé, fichier accepté, ligne retirée, ligne acceptée, champ mis à null, …).

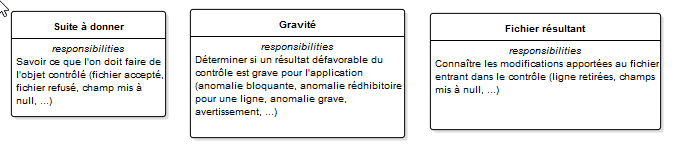
Par ailleurs, si dans une ligne d'un fichier HIT un champ comme le "Département" n'est pas renseigné, la section de trafic correspondante à cette ligne est inutilisable puisque l'on ne pourra pas la localiser. Le champ Département est un champ "**rédhibitoire**". La MOA a stipulé une *règle de gestion* précisant qu'une telle ligne devait être éliminée. Le fichier en sortie du contrôle peut donc être différent du fichier en entrée puisque des lignes peuvent être éliminées.

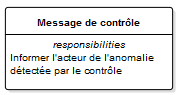
On voit donc apparaître le concept de **Gravité** d'un contrôle (anomalie bloquante, anomalie rédhibitoire pour une ligne, anomalie grave, avertissement, …). Il existe une relation entre la gravité, l'objet contrôlé et la suite à donner puisque l'on peut imaginer qu'un contrôle défavorable d'une forte gravité sur un champ rédhibitoire peut entraîner la suppression d'une ligne, la mise à null d'un champ, …

On voit également apparaître le concept de **Fichier résultant** d'un contrôle. Un fichier peut par exemple comporter moins de lignes en sortie d'un contrôle sur le champ Département qu'en entrée puisque l'on peut avoir retiré les lignes avec le Département non renseigné.

En outre, on constate que tout contrôle doit générer un **Message** de contrôle à l'intention des acteurs. La MOA a donc un *exigence fonctionnelle implicite que tout contrôle génère un message de contrôle*.

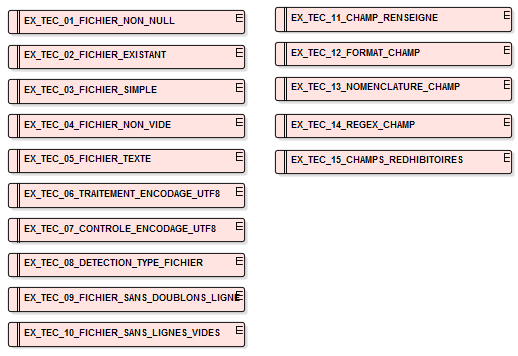
Concepts capturés :



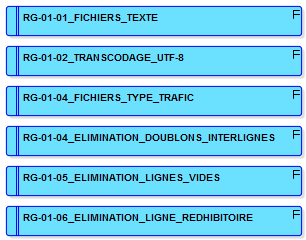


Exigences capturées :





Règles de Gestion capturées :

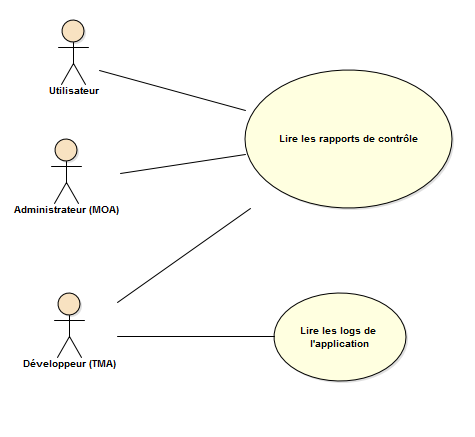


Résumé tabulaire des Règles de gestion capturées :

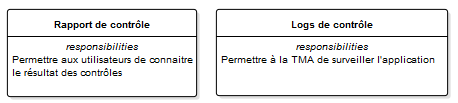


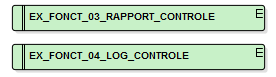
### Identifier les acteurs et imaginer comment l'application devra rendre compte des contrôles (rapports de contrôle, logs)

* L'**Utilisateur** de l'application informatique aura *besoin d'un rapport de contrôle* lui indiquant pourquoi une ressource (fichier soumis en entrée) est défectueuse.
* De même l'**Administrateur** de l'application informatique (**M**aîtrise d'**O**uvr**A**ge MOA) aura besoin d'accéder à ces rapports de contrôle pour aider les utilisateurs à mieux utiliser l'application.
* Enfin, un **Développeur** tiers (**T**ierce **M**aintenance **A**pplicative TMA) aura besoin de savoir comment les rapports de contrôle ont été conçus et sont générés. Il aura également *besoin des logs* de l'application pour en améliorer l'utilisation.



#### Concepts et exigences mis en évidence





## Identifier les concepts

### Identifier les concepts nécessaires à minima pour l'acteur le plus exigeant (Quand-Qui-Quoi-Qu'est ce - Que-Quel-Où)

L'objectif est d'identifier grossièrement les concepts permettant de qualifier le concept Contrôle.

Un **Utilisateur** sait quel fichier il vient d'utiliser dans l'application. L'**Administrateur** ou le **Développeur** auront probablement besoin d'examiner les rapports de contrôles à postériori. Ils auront donc besoin de renseignements leur permettant d’identifier quel objet contrôlé (fichier, ligne du fichier, …) a déclenché tel contrôle…

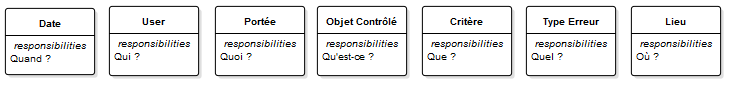
Il y a donc une *exigence fonctionnelle de pouvoir parfaitement identifier un contrôle* à postériori pour l'**Administrateur** et le **Développeur**. La suite de questions **Quand-Qui-Quoi-Qu'est ce-Que-Quel-Où** est une bonne piste pour commencer à trouver l'identifiant.

1. **Quand** le contrôle a-t-il eu lieu ? : concept **Date.** Date d'exécution du contrôle.
2. **Qui** a déclenché le contrôle ? : concept **User.** Utilisateur qui a déclenché le contrôle.
3. **Quoi** – le contrôle contrôlait quoi ? Un fichier ?, un champ d'une ligne du fichier ? : concept **Portée** (aussi appelé Niveau)**.** Type d'objet(s) sur lequel porte le contrôle.
4. **Qu'est-ce** – sur quel objet le contrôle s'applique-t-il ? le fichier DIRA\_2014 ? Le champ département de la 3ème ligne du fichier DIRE\_2013 ? : concept **Objet**. L'objet sur lequel porte le contrôle.
5. **Que** détecte le contrôle ? : concept **Critère.** Le critère que doit vérifier l'objet contrôlé. Par exemple, "le fichier ne doit comporter aucun caractère indésirable" ou "une ligne dans un HIT doit comporter 520 caractères".
6. **Quel** type d'erreur le contrôle recherche-t-il ? : concept **Type**. Le type de l'erreur contrôlée comme par exemple "taille", "règle de gestion", "renseigné", …
7. **Où** le contrôle a-t-il eu lieu ? (contrôle de surface, métier …) : concept **Lieu**

Par exemple :



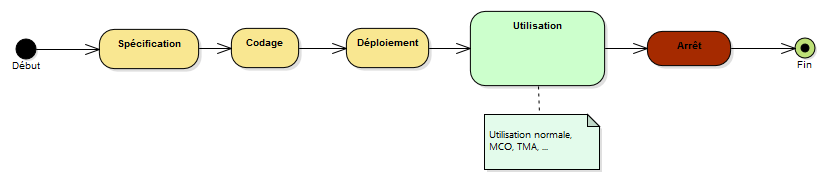
#### Concepts et exigences mis en évidence





### Identifier les concepts liés au cycle de vie d’une application

On peut modéliser le cycle de vie d’une application informatique comme suit :



Les phases de codage, de déploiement et d’utilisation de l’application engendrent des besoins et exigences en sus de la phase de spécification.

#### Imaginer le codage de l’application en équipe

Le travail en équipe nécessite de se mettre d'accord au préalable sur le comportement attendu de l'application.

Il faut s'entendre sur ce que l'on entend par "un contrôle a répondu favorablement". Par exemple, que conclure si le Contrôle du contenu du fichier (fichier pas vide) a retourné true ? Le fichier était-il vide ou pas ?

Il faut donc une règle de gestion :



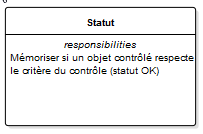
Définition : Un contrôle est dit "favorable" si l'objet contrôlé remplissait le critère de contrôle.

Un contrôle favorable (qui a détecté que l'objet remplissait le critère) doit répondre true dans l'application.

Par exemple le résultat du contrôle "Pas de doublons" à true signifie que le fichier ne comporte pas de lignes en doublon.

Par ailleurs, la phrase "le contrôle a répondu favorablement (true)" fait apparaître un nouveau concept : le **Statut** du contrôle après son exécution.

* Lorsqu'un contrôle sur un objet répond favorablement (true), cela signifie que l'objet respectait le critère du contrôle (champ département renseigné par exemple). Le statut du contrôle est donc favorable (OK)
* Lorsqu'un contrôle sur un objet répond défavorablement (false), cela signifie que l'objet ne respectait pas le critère du contrôle (champ département renseigné par exemple). Le statut du contrôle est donc défavorable (KO).



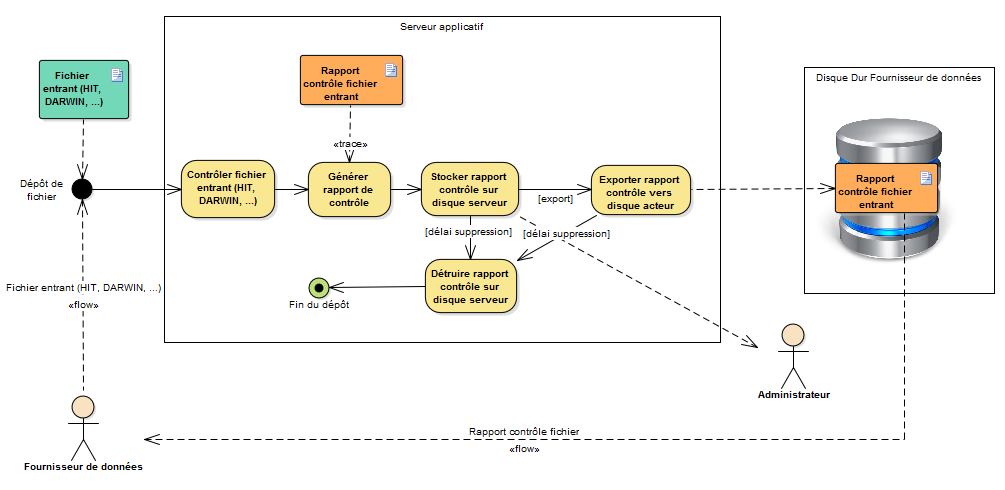
#### Imaginer le déploiement de l’application

#### Imaginer la « vie réelle de l’application »

La question est de savoir comment les acteurs accèderont concrètement aux rapports de contrôle et comment ils bénéficieront de ce rapport.

Un Acteur "Fournisseur de données" aura sans doute besoin d'importer sur son propre disque dur le rapport de contrôle du fichier qu'il vient de déposer dans l'application afin de pouvoir l'étudier. L'application doit donc disposer de ce rapport de contrôle sur le serveur applicatif et permettre de l'exporter vers le Fournisseur de données.

### Simuler le déclenchement des contrôles par un fournisseur de données



### Imaginer la gestion des contrôles par l'Administrateur

L'administrateur de l'application peut être amené lors de la phase d'utilisation normale de l'application à activer/désactiver certains contrôles (par exemple si des contrôles de faible gravité polluent les rapports de contrôle). Les contrôles doivent donc être *individuellement* *paramétrables* pour s'appliquer ou non. On voit apparaître le concept d'**Activité** d'un contrôle.

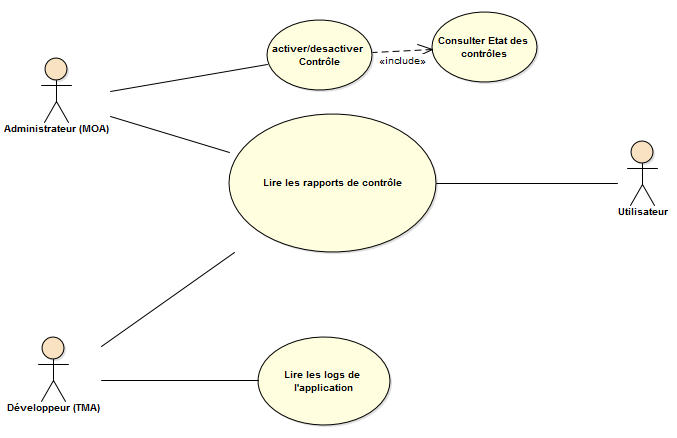
La MOA a donc une *exigence fonctionnelle de contrôles paramétrables*.

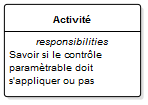
La MOA a stipulé une *règle de gestion* précisant qu'un contrôle désactivé ne devait pas s'appliquer et émettre de rapport.

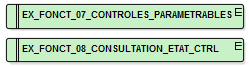
En pratique, l'Administrateur aura besoin d'une aide logicielle pour connaître la liste des contrôles et savoir quels sont les contrôles activés avant de pouvoir activer/désactiver un contrôle.

La MOA a donc également une *exigence fonctionnelle de consultation de l'état des contrôles*.

#### Concepts, exigences et RG mis en évidence





### Imaginer comment les acteurs utiliseront les rapports de contrôle (tris, requêtes, …)

### Lister les questions types que se posera la MOA

### Capturer les concepts

## Synthétiser

### Synthèse des acteurs retenus

### Synthèse des exigences retenues

### Synthèse du modèle conceptuel retenu

## Affiner et Structurer les concepts

## Etudier l'enchaînement des contrôles

## Etudier la réutilisation des contrôles

# Conception des contrôles

## "jouer" des contrôles

# Administration des contrôles dans l'application

# Capitalisation de la méthode