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

Table des matières

[1 Objectif de l'étude 3](#_Toc451849118)

[2 Analyse du concept Contrôle 4](#_Toc451849119)

[2.1 Dégrossir le problème (Collecter les concepts, besoins, exigences et règles de gestion) 4](#_Toc451849120)

[2.1.1 Collecter les concepts, besoins, exigences et règles de gestion (RG) liés aux contrôles auprès de la MOA 4](#_Toc451849121)

[2.1.2 Appliquer les règles de l’art 5](#_Toc451849122)

[2.1.3 Identifier quelques cas réels de contrôles 5](#_Toc451849123)

[2.1.4 Identifier les acteurs et imaginer comment l'application devra rendre compte des contrôles (rapports de contrôle, logs) 11](#_Toc451849124)

[2.2 Simuler l'utilisation des contrôles 12](#_Toc451849125)

[2.2.1 Identifier les concepts nécessaires à minima pour l'acteur le plus exigeant (Quand-Qui-Quoi-Qu'est ce - Que-Quel-Où) 12](#_Toc451849126)

[2.2.2 Identifier les concepts liés au cycle de vie d’une application 14](#_Toc451849127)

[2.2.3 Simuler le déclenchement des contrôles par un fournisseur de données 18](#_Toc451849128)

[2.2.4 Simuler la gestion des contrôles par l'Administrateur 19](#_Toc451849129)

[2.2.5 Simuler l'utilisation des rapports de contrôle (tris, requêtes, …) 20](#_Toc451849130)

[2.2.6 Lister les questions types que se posera la MOA 21](#_Toc451849131)

[2.3 Synthétiser les acteurs, concepts, exigences et RG capturés 22](#_Toc451849132)

[2.3.1 Synthèse des acteurs capturés 22](#_Toc451849133)

[2.3.2 Synthèse des concepts capturés 24](#_Toc451849134)

[2.3.3 Synthèse des exigences capturées 24](#_Toc451849135)

[2.3.4 Synthèse des RG capturées 28](#_Toc451849136)

[2.4 Affiner et Structurer les concepts 30](#_Toc451849137)

[2.4.1 Concept Date 31](#_Toc451849138)

[2.4.2 Concept User 32](#_Toc451849139)

[2.4.3 Concept Portée 33](#_Toc451849140)

[2.4.4 Concept Objet Contrôlé 34](#_Toc451849141)

[2.4.5 Concept Critère 35](#_Toc451849142)

[2.4.6 Concept Type d'Erreur 36](#_Toc451849143)

[2.4.7 Concept Lieu 37](#_Toc451849144)

[2.4.8 Concept Activité 38](#_Toc451849145)

[2.4.9 Concept Statut 39](#_Toc451849146)

[2.4.10 Concept Gravité 40](#_Toc451849147)

[2.4.11 Concept Suite A Donner 41](#_Toc451849148)

[2.4.12 Concept Fichier Résultant 42](#_Toc451849149)

[2.4.13 Concept Règle de Gestion 44](#_Toc451849150)

[2.4.14 Concept Règle de l'Art 45](#_Toc451849151)

[2.4.15 Concept Rapport de Contrôle 46](#_Toc451849152)

[2.4.16 Concept Log de Contrôle 47](#_Toc451849153)

[2.4.17 Concept Message de Contrôle 48](#_Toc451849154)

[2.5 Bâtir le modèle conceptuel des Contrôles 48](#_Toc451849155)

[2.5.1 Etablir les relations inter-concepts (autres que Contrôle) 48](#_Toc451849156)

[2.5.2 Etudier l'enchaînement des contrôles 48](#_Toc451849157)

[2.5.3 Modèle conceptuel des contrôles 49](#_Toc451849158)

[2.6 Tester le modèle conceptuel des Contrôles 49](#_Toc451849159)

[3 Conception des contrôles 49](#_Toc451849160)

[3.1 Extraire les attributs et méthodes livrées par le modèle conceptuel 49](#_Toc451849161)

[3.2 Factoriser au maximum le code (abstractions) 49](#_Toc451849162)

[3.3 Appliquer les règles de codage (Design Patterns) 49](#_Toc451849163)

[3.4 "jouer" des contrôles 49](#_Toc451849164)

[3.5 Bâtir le modèle de classe des contrôles 50](#_Toc451849165)

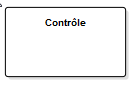
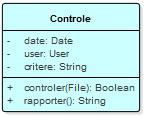
[4 Administration des contrôles dans l'application 50](#_Toc451849166)

[5 Capitalisation de la méthode 50](#_Toc451849167)

# Objectif de l'étude

L'objectif de la présente étude est de **modéliser les concepts qualifiant 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.

On tente d'élaborer une **méthode opérationnelle** pour *capturer* les **concepts**, **besoins**, **exigences** et **règles de gestion** auxquels doivent satisfaire les contrôles dans une application informatique.

L'idée est de définir le périmètre couvert par les contrôles, d'**analyser** les concepts capturés, puis de **concevoir** des classes de contrôles répondant aux besoins capturés.

# Analyse du concept Contrôle

L'objectif du chapitre "Analyse du concept Contrôle" est de **dégager les éléments essentiels d'un contrôle** au vu des besoins, exigences, règles de gestion ainsi que de l'utilisation ultérieure des contrôles par les divers acteurs.

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

Objectif : L'objectif de cette partie "Dégrossir…" est de comprendre les premiers besoins et exigences *en matière de contrôle* dans une application informatique. Il s'agit de déterminer le **périmètre couvert** par les contrôles (équivalent de la phase d'inception dans le **R**ational **U**nified **P**rocess RUP).

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

Objectif : capturer les concepts, besoins, exigences et règles de gestion formulés par la **M**aîtrise d'**O**uvr**A**ge (MOA).

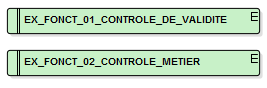
Toute application informatique a besoin de **contrôler** la validité des ressources qu'elle accepte en entrée ainsi que la qualité de ses livrables. La **M**aîtrise d'**O**uvr**A**ge (MOA) a donc une *exigence fonctionnelle de contrôle de validité* des fichiers entrants.

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

Objectif : capturer les concepts, besoins, exigences et règles de gestion implicites provenant des 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



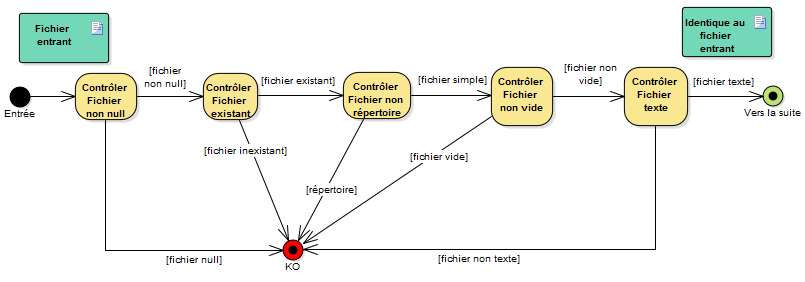


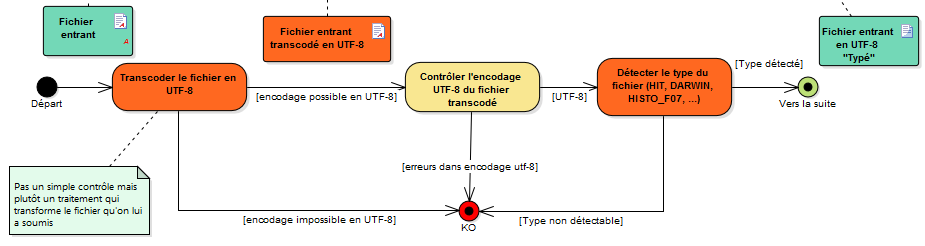
### Identifier quelques cas réels de contrôles

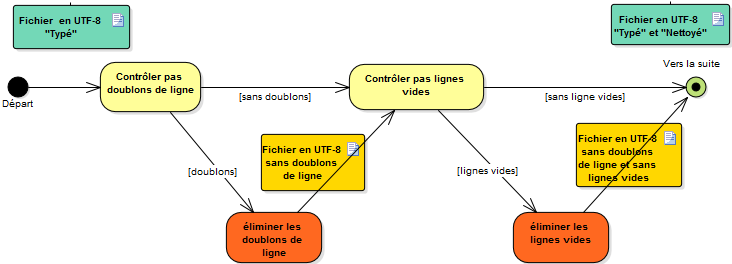
Objectif : capturer les concepts, besoins, exigences et règles de gestion provenant de l'étude de quelques cas réels.

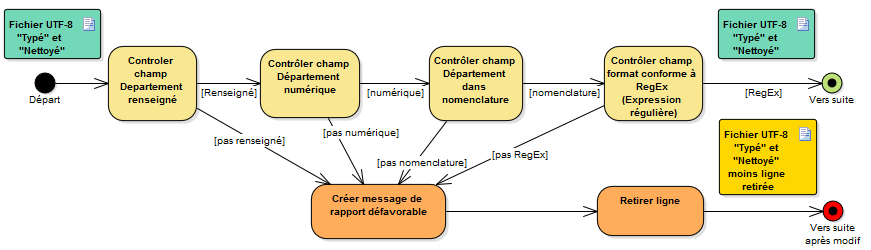
**Détecter quelques contrôles** et traitements que l'application devra obligatoirement exécuter. L'idée est de dégrossir le problème et de se faire une idée de ce qu'est le concept de contrôle. On applique notamment des **règles de l’art** (comme par exemple, il est toujours inutile d’accepter en entrée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

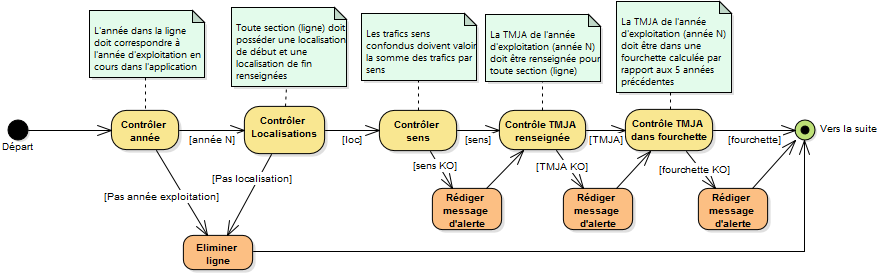
Par exemple, en analysant le début du traitement d'un fichier dans l'application TraficWeb :











#### 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 |
| 20 | Métier | Champ | Règle de gestion | Contrôle métier sur l'année qui doit valoir l'année d'exploitation |
| 21 | Métier | Inter-champs | Règle de gestion | Contrôle métier sur Les localisations de début et de fin d'une section qui doivent être renseignées. |
| 22 | Métier | Inter-Lignes | Règle de gestion | Contrôle métier sur les sens des trafics |

#### 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, …).

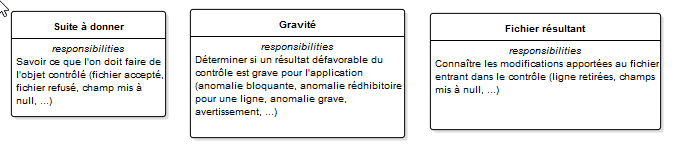
Par ailleurs, si dans une ligne d'un fichier HIT un champ comme le "Département" n'est pas renseigné, la section de trafic correspondante à cette ligne est inutilisable puisque l'on ne pourra pas la localiser. Le champ Département est un champ "**rédhibitoire**". La MOA a stipulé une *règle de gestion* 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.

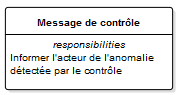
On voit donc apparaître le concept de **Gravité** d'un contrôle (anomalie bloquante, anomalie rédhibitoire pour une ligne, anomalie grave, avertissement, …). Il existe une relation entre la gravité, l'objet contrôlé et la suite à donner puisque l'on peut imaginer qu'un contrôle défavorable d'une forte 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 (y compris les traitements) doit générer un **Message** de contrôle à l'intention des acteurs. La MOA a donc un *exigence fonctionnelle implicite que tout contrôle ou traitement génère un message de contrôle*.

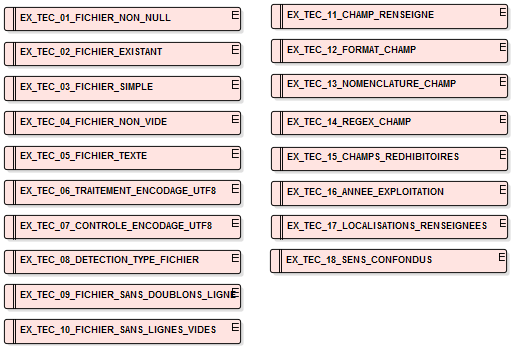
Concepts capturés :





Exigences capturées :





Règles de Gestion capturées :



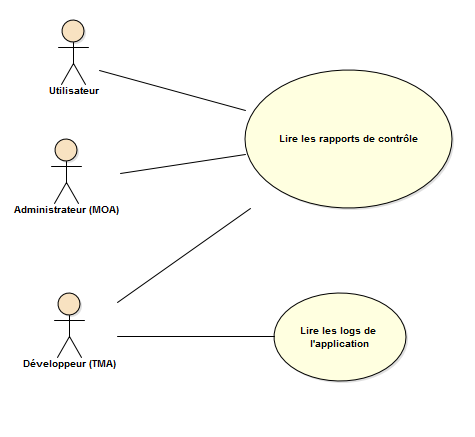
Résumé tabulaire des Règles de gestion capturées :



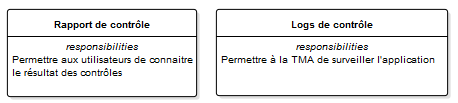
### Identifier les acteurs et imaginer comment l'application devra rendre compte des contrôles (rapports de contrôle, logs)

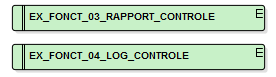
Objectif : capturer les concepts, besoins, exigences et règles de gestion provenant de l'étude des besoins des différents acteurs.

* 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





## Simuler l'utilisation des contrôles

Objectif : L'objectif de cette phase "simuler l'utilisation des contrôles" est de comprendre ce que "doit savoir" un contrôle pour pouvoir être utilisé dans de bonnes conditions par les différents acteurs.

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

Objectif : identifier grossièrement les concepts, besoins, exigences et règles de gestion permettant d'identifier un concept Contrôle.

Un **Utilisateur** (fournisseur de données) qui vient de déposer un fichier sait quel fichier il vient d'utiliser dans l'application. Il n'aura donc aucun problème pour identifier à quel objet contrôlé (fichier) se rapporte le rapport de contrôle que l'application vient de générer.

En revanche, 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, champ…) a déclenché tel contrôle…

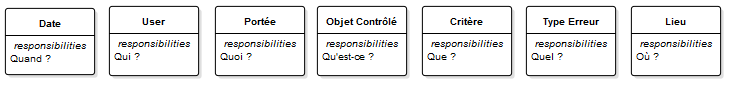
Il y a donc une *exigence fonctionnelle de pouvoir parfaitement identifier un contrôle* à postériori pour l'**Administrateur** et le **Développeur**. La suite de questions **Quand-Qui-Quoi-Qu'est ce-Que-Quel-Où** est une bonne piste pour commencer à trouver l'identifiant.

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 comporte aucun caractère indésirable" ou "une ligne dans un HIT comporte exactement 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, contrôle métier …) : concept **Lieu**

Par exemple :



#### Concepts et exigences mis en évidence

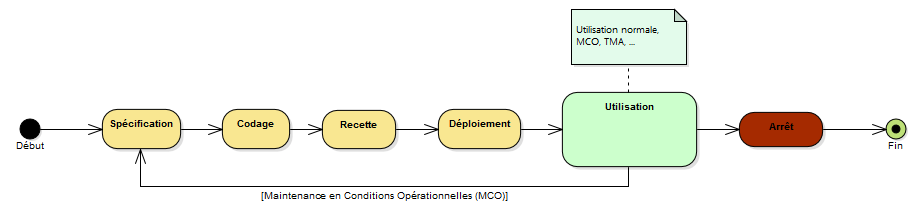




### Identifier les concepts liés au cycle de vie d’une application

Objectif : identifier les concepts, besoins, exigences et règles de gestion liés aux contrôles provenant de l'étude du cycle de vie d'une application informatique.

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



Les phases de codage, de recette (tests), 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

Objectif : identifier les concepts, besoins, exigences et règles de gestion liés aux contrôles provenant de l'étude du codage d'une application informatique en équipe.

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

Il faut se mettre d'accord 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 appliquées par tous les développeurs de l'équipe :



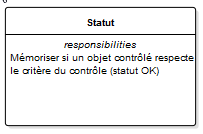
Définition : Un contrôle est dit "favorable" si l'objet contrôlé *remplissait* le critère de contrôle.

RG-100\_TRUE\_SI\_CONTROLE\_FAVORABLE : 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.
* Autre exemple : le résultat du contrôle "Contenu du fichier pas vide" à true signifie que le fichier n'est pas vide.

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 (critère = 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 (critère = champ département renseigné par exemple). Le statut du contrôle est donc défavorable (KO).



#### Imaginer la recette de l'application

Objectif : identifier les concepts, besoins, exigences et règles de gestion liés aux contrôles provenant de la simulation des futures recettes fonctionnelles et techniques de l'application informatique.

La recette d'une application informatique suppose qu'un acteur "**Recetteur**" puisse fabriquer des jeux d'essais et les jouer dans l'application afin de tester les contrôles.

Il est donc indispensable que le recetteur ait un *accès spécifique aux contrôles* de l'application afin de pouvoir tester ses jeux d'essais.

Le **Recetteur fonctionnel** doit pouvoir préparer des fichiers particuliers, les déposer dans l'application et vérifier que les contrôles voulus se sont bien déclenchés et que l'application délivre le bon rapport de contrôle.

Le **Recetteur technique** doit pouvoir tester chaque classe de contrôle individuellement (tests unitaires JUnit) et vérifier qu'il n'y a pas de régression entre les versions successives de l'application.



#### Imaginer le déploiement de l’application

Objectif : identifier les concepts, besoins, exigences et règles de gestion liés aux contrôles provenant de l'étude du déploiement de l'application informatique.

Si l'application informatique est une application web, elle est susceptible d'être hébergée par un centre serveur.

Dans le cas où les contrôles nécessitent des ressources (fichiers externalisés properties, fichiers de nomenclatures pour certains champs, fichiers de description des champs pour les HIT, DARWIN, …), il est recommandé de permettre au centre serveur de choisir où il stockera ces ressources.

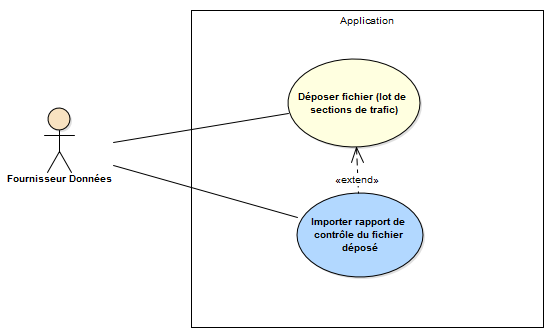
L'application doit donc prévoir le paramétrage des chemins des ressources nécessaires aux contrôles (Exigence fonctionnelle de l'application).

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

Objectif : identifier les concepts, besoins, exigences et règles de gestion liés aux contrôles provenant de la simulation de l'utilisation réelle de l'application informatique.

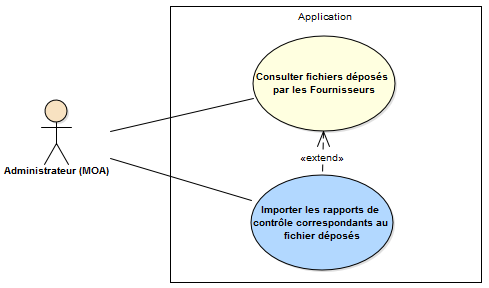
La question est de savoir comment les acteurs accèderont concrètement aux rapports de contrôle et comment ils bénéficieront de ce rapport.

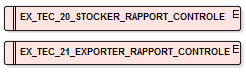
Un Acteur "**Fournisseur de données**" (**D**irection **I**nterdépartementale des **R**outes (DIR), **S**ociété **C**oncessionnaire d'**A**utoroute (SCA), …) aura sans doute besoin d'importer sur son propre disque dur le rapport de contrôle du fichier qu'il vient de déposer dans l'application afin de 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*.



Un acteur "**Administrateur (MOA)**" aura besoin de surveiller la qualité des fichiers déjà déposés par des fournisseurs de données. Il aura donc besoin de consulter à postériori les rapports de contrôle correspondants aux fichiers déposés.

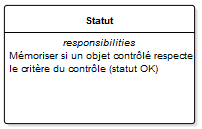
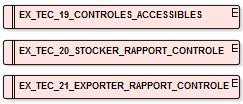
Il souhaitera sans doute importer ces rapports de contrôle sur son propre disque dur afin de les agréger, trier, comparer, …





Par ailleurs, le Fournisseur de Données qui a exporté sur son disque dur le rapport de contrôle dispose du fichier original de données trafic qu'il a soumis et du rapport de contrôle. Or, le fichier de données trafic peut être modifié (lignes retirées, champs mis à null, …) lors du processus de trafic. L'application doit donc *mémoriser les numéros de ligne dans les fichiers originaux* et les utiliser dans les rapports de contrôle afin de simplifier le travail du fournisseur.

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

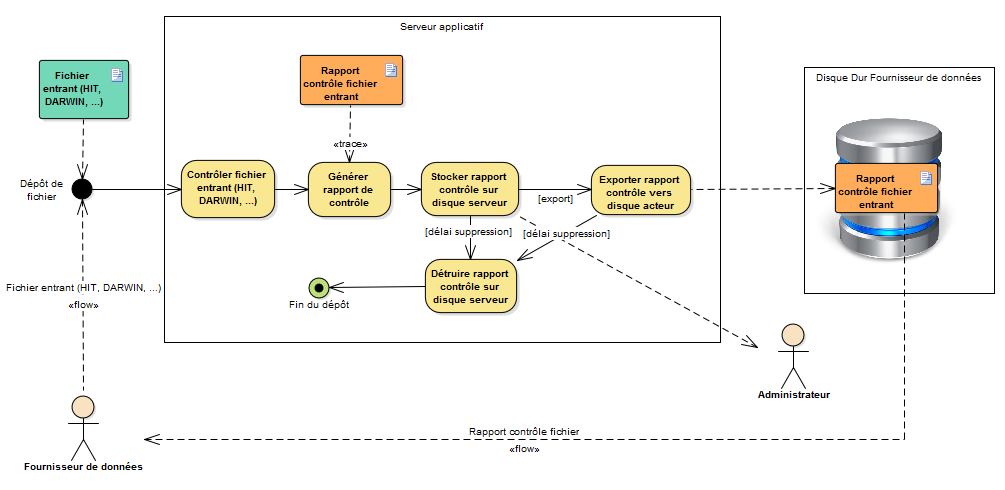
 

### Simuler le déclenchement des contrôles par un fournisseur de données

Objectif : identifier les concepts, besoins, exigences et règles de gestion liés aux contrôles provenant de la simulation du dépôt d'un fichier dans l'application informatique.

Le Fournisseur de données (DIR, SCA, …) dépose un fichier (HIT, DARWIN, HISTONAT, ...) dans l'application TraficWeb. TraficWeb réalise alors le processus suivant :

Import du fichier déposé > Contrôle > Génération du rapport de contrôle > Stockage du rapport sur le serveur



Les acteurs peuvent ensuite importer les rapports de contrôle sur leur propre disque dur comme vu précédemment. L'Administrateur MOA ira chercher directement le rapport de contrôle sur le serveur.

Le problème est que le stockage "sans limite" des rapports de contrôle sur le serveur peut conduire à un problème de taille de stockage.

Il est nécessaire qu'un acteur du type "**Administrateur Applicatif**" puisse s'assurer que le stockage des rapports de contrôle n'occupe pas un volume excessif sur le serveur. Le cas échéant, l'Administrateur Applicatif doit pouvoir *retirer manuellement des rapports de stockage*.

On peut également imaginer que l'application "sache" *automatiquement écraser les rapports de contrôle* hors d'une période déterminée par la MOA (par exemple, l'année d'exploitation).

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



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

Objectif : identifier les concepts, besoins, exigences et règles de gestion liés aux contrôles provenant de la simulation de la gestion des contrôles "lors de la vie réelle" par l'Administrateur MOA dans l'application informatique.

L'**Administrateur (MOA)** 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.

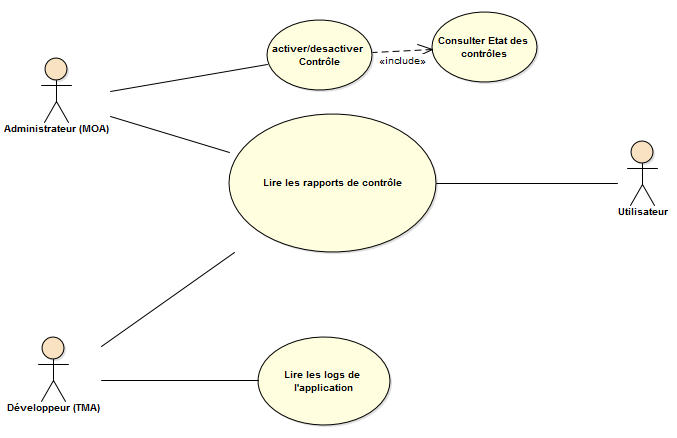
La MOA a donc une *exigence fonctionnelle de contrôles paramétrables*.

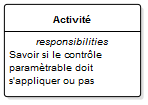
La MOA a stipulé une *règle de gestion* précisant qu'un contrôle désactivé ne devait pas s'appliquer et émettre de rapport.

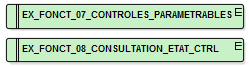
En pratique, l'Administrateur MOA aura besoin d'une aide logicielle pour connaître la liste des contrôles et savoir quels sont les contrôles activés avant de pouvoir activer/désactiver un contrôle.

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

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





### Simuler l'utilisation des rapports de contrôle (tris, requêtes, …)

Objectif : identifier les concepts, besoins, exigences et règles de gestion liés aux contrôles provenant de la simulation de la gestion des contrôles par l'Administrateur MOA dans l'application informatique.

L'**Administrateur (MOA)** aura sans doute besoin de vérifier les rapports de contrôles des derniers lots de sections de trafic déposés par les **Fournisseurs de Données** avant de décider de consolider les données et de générer un fichier national annuel HISTONAT\_F07.

Il doit pouvoir agréger les données trafic provenant de chaque Fournisseur de Données (exigence fonctionnelle de l'application) pour procéder à une consolidation nationale.

Il doit pouvoir également *agréger les rapports de contrôle* afin d'avoir une vue globale de la qualité des données.

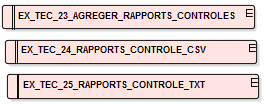
Comment peut-il utiliser ce rapport de contrôle agrégé ?

Une bonne solution semble être de présenter ce rapport agrégé sous forme de tableau *lisible par n'importe quel tableur*. Le format csv semble donc tout indiqué puisque – sous réserve que l'on respecte un certain ordre des colonnes – il permet l'agrégation des données comme leur vision sous forme de tableau (Excel, Calc, …). Il y a donc une *exigence fonctionnelle de consolidation des rapports* et une *exigence technique de présentation des lignes de rapport au format csv*.

L'Administrateur (MOA) pourra ensuite faire ses tris, ses analyses croisées, compter combien il y a de messages de contrôle défavorable pour telle DIR, rechercher une ligne de rapport particulière dans le tableur…

Si l'application est de type "Desktop" (avec des fenêtres et tournant directement sur la machine de l'acteur) et pas une application Web, les zones de texte ne se prêtent pas à l'affichage de lignes csv. Il faut donc prévoir de présenter également les messages de contrôle sous forme textuelle (pas csv) pour l'affichage des messages de contrôle dans des zones de texte. Il y a donc également une *exigence technique de présentation des lignes de rapport au format textuel*.

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

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

Objectif : identifier les concepts, besoins, exigences et règles de gestion liés aux contrôles provenant de la simulation des questions posées par l'Administrateur MOA à l'application informatique.

La MOA aura besoin d'étudier les rapports de contrôle pour :

* Evaluer la qualité des données trafic.
* Déterminer les objets (fichiers, lignes, champs, …) qui posent problème.
* Superviser le travail des Fournisseurs de données (DIR, SCA, …)
* Prioriser ses interventions pour améliorer la qualité des données trafic.

On peut dégager quelques questions type :

#### Quelle est la qualité d'ensemble d'une année d'exploitation ?

La MOA doit donc pouvoir consolider les rapports de contrôle d'une année d'exploitation et bénéficier d'un *indicateur de qualité*. Par exemple, **combien de lignes comportent au moins une anomalie rédhibitoire par rapport au total des lignes**.

Le concept de Statut du contrôle (OK/KO) déjà évoqué est un bon moyen pour créer cet indicateur.

#### Quelle est la tendance de la qualité ?

La MOA doit donc pouvoir comparer les rapports de contrôle consolidés entre plusieurs années d'exploitation. Le fait de disposer des rapports de contrôle sous forme de tableau (csv) répond à ce besoin dès lors que le tableau contient le champ année d'exploitation et dès lors que la MOA stocke les rapports annuels.

#### Quelles sont les Fournisseurs de données (DIR, …) qui posent problème ?

La MOA doit pouvoir comparer les rapports de contrôle consolidés annuellement par DIR par exemple.

Le fait de disposer des rapports de contrôle sous forme de tableau (csv) répond à ce besoin dès lors que le tableau contient le champ Fournisseur de données.

#### Quels sont les champs qui posent problème ?

La MOA souhaitera savoir par exemple quels sont les champs très peu renseignés. Elle souhaitera également savoir quels sont les types d'erreur (non renseigné, mauvais format, hors nomenclature, hors Expression Régulière (RegEx), ...) les plus fréquentes pour un champ donné.

Le fait de disposer des rapports de contrôle sous forme de tableau (csv) répond à ce besoin dès lors que le tableau contient le champ contrôlé et le type d'erreur diagnostiqué.

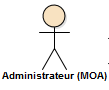
#### Concepts, exigences et RG mis en évidence

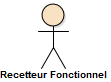
Les concepts et exigences capturés précédemment sont suffisants ici.

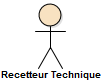
## Synthétiser les acteurs, concepts, exigences et RG capturés

Objectif : l'objectif de cette partie est de faire à ce stade une synthèse des divers acteurs, concepts, exigences et RG précédemment capturés.

### Synthèse des acteurs capturés

#### Fournisseur de Données

Rôle tenu par tout acteur qui dépose un fichier (lot de sections de trafic) dans l'application.

A priori, il dépose des fichiers HIT ou DARWIN. C'est souvent un agent en **D**irection **I**nterdépartementale des **R**outes (DIR) qui dépose les trafics pour sa DIR. Mais on peut imaginer également le dépôt d'un fichier HISTONAT\_F07 (fichier consolidé national annuel des trafics) par la MOA pour effectuer des contrôles dessus par exemple.

#### Administrateur MOA

Rôle tenu par tout acteur qui supervise le travail déposé par les fournisseurs de données.

En particulier, il a besoin de consulter les rapports de contrôles correspondants aux fichiers déposés par les fournisseurs pour évaluer la qualité des données. Il peut également paramétrer les contrôles (actif/inactif). Il peut également déposer à postériori dans l'application un fichier annuel national consolidé HISTONAT\_F07 pour en vérifier la qualité.

#### Administrateur Applicatif

Rôle tenu par tout acteur chargé par la MOA d'administrer techniquement l'application (vidage des logs, vidage du stockage des rapports de contrôle, gestion des droits applicatifs, arrêt du serveur, …).

#### Utilisateur Consultant

Rôle tenu par tout acteur qui se borne à lire les rapports de contrôle.

#### Développeur (TMA)

Rôle tenu par tout acteur chargé de développer et maintenir l'application informatique. Il a besoin de consulter les rapports de contrôle ainsi que les logs générés par les contrôles. Il assure la **T**ierce **M**aintenance **A**pplicative (TMA) et le **M**aintien en **C**onditions **O**pérationnelles (MCO) de l'application.

#### Recetteur Fonctionnel

Rôle tenu par tout acteur chargé de la recette fonctionnelle de l'application. Il doit pouvoir "jouer" des jeux d'essais prédéterminés pour solliciter les contrôles de l'application.

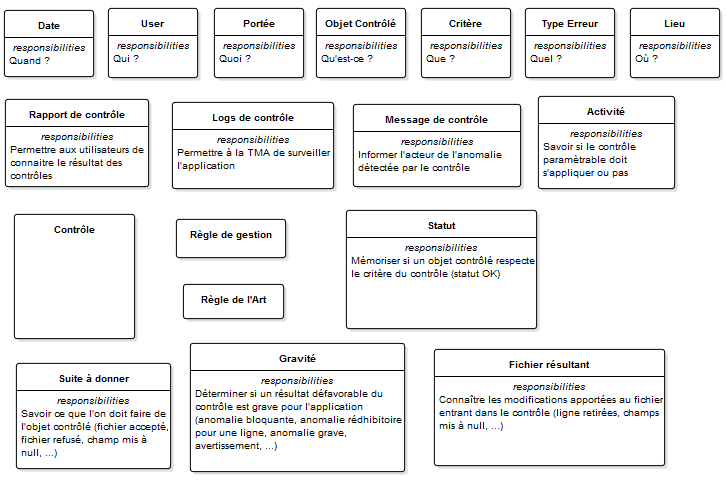
#### Recetteur Technique

Rôle tenu par tout acteur chargé de la recette technique de l'application. Il doit pouvoir "jouer" des tests unitaires (JUnit) et tester la non-régression de l'application entre différentes versions.

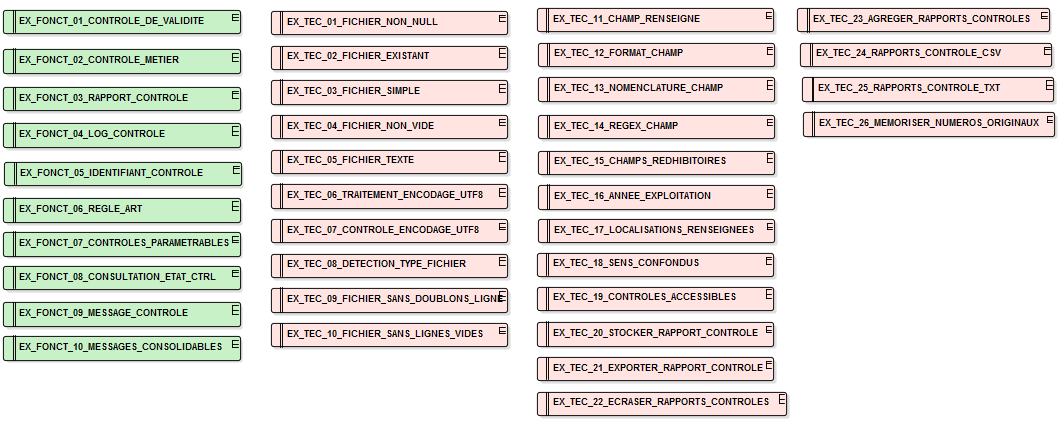
#### Centre Serveur

Rôle tenu par tout acteur chargé d'héberger l'application informatique, et par conséquent ses contrôles. Il a besoin de savoir où placer les ressources (properties, fichiers de nomenclature, fichiers de description, ...) utilisées par les contrôles.

### Synthèse des concepts capturés

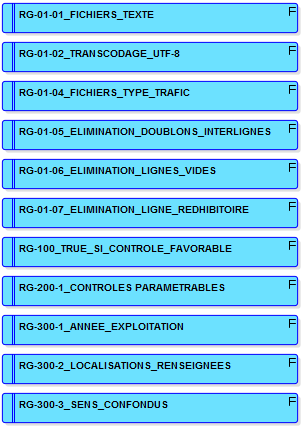


### Synthèse des exigences capturées



| **DEMANDEUR** | **IDENTIFIANT DE L'EXIGENCE** | **INTITULE** | **Description** | **Catégorie de l'exigence** |
| --- | --- | --- | --- | --- |
| MOA | EX\_FONCT\_01\_CONTROLE\_DE\_VALIDITE | Contrôle de la validité des fichiers en entrée | L'application cible doit contrôler la validité des fichiers acceptés en entrée. | Fonctionnelle |
| MOA | EX\_FONCT\_02\_CONTROLE\_METIER | Contrôle des données métier | L'application cible doit contrôler que les données métier satisfont à des règles de gestion | Fonctionnelle |
| MOA | EX\_FONCT\_03\_RAPPORT\_CONTROLE | Tout contrôle doit générer un rapport de contrôle | L'application cible doit fournir des rapports de contrôle expliquant le résultat de tous ses contrôles et traitements | Fonctionnelle |
| MOA | EX\_FONCT\_04\_LOG\_CONTROLE | Tout contrôle doit envoyer un message dans un Log applicatif | L'application cible doit fournir des logs applicatifs pour les contrôles | Fonctionnelle |
| MOA | EX\_FONCT\_05\_IDENTIFIANT\_CONTROLE | Tout contrôle doit être identifié | L'application cible doit identifier les contrôles afin de pouvoir les retrouver, les trier, … | Fonctionelle |
| MOA | EX\_FONCT\_06\_REGLE\_ART | Les contrôles doivent respecter les règles de l'art | L'application cible doit respecter les règles de l'art | Fonctionnelle |
| MOA | EX\_FONCT\_07\_CONTROLES\_PARAMETRABLES | Tout contrôle doit pouvoir être activé/désactivé | L'application cible doit mettre à disposition de l'Administrateur un dispositif de paramétrage individuel des contôles | Fonctionnelle |
| MOA | EX\_FONCT\_08\_CONSULTATION\_ETAT\_CTRL | Tout contrôle doit savoir si il est actif ou pas | L'application cible doit mettre à disposition de l'Administrateur un dispositif de consultation du paramétrage individuel des contôles | Fonctionnelle |
| MOA | EX\_FONCT\_09\_MESSAGE\_CONTROLE | Tout contrôle doit générer un message de contrôle à l'attention des acteurs | L'application cible doit générer un message de contrôle pour tout contrôle appliqué | Fonctionnelle |
| MOA | EX\_FONCT\_10\_MESSAGES\_CONSOLIDABLES | Les messages de contrôles doivent être consolidables (géographiquement au niveau national et temporellement sur une année d'exploitation) | L'application cible doit générer des messages de contrôle consolidables (géographiquement au niveau national et temporellement sur une année d'exploitation) | Fonctionnelle |
| MOE | EX\_TEC\_01\_FICHIER\_NON\_NULL | Un fichier en entrée ne doit pas être null | L'application cible doit contrôler que le fichier en entrée n'est pas null | Technique |
| MOE | EX\_TEC\_02\_FICHIER\_EXISTANT | Un fichier en entrée doit être existant | L'application cible doit contrôler que le fichier en entrée est existant | Technique |
| MOE | EX\_TEC\_03\_FICHIER\_SIMPLE | Un fichier en entrée ne doit pas être un répertoire | L'application cible doit contrôler que le fichier en entrée n'est pas un répertoire | Technique |
| MOE | EX\_TEC\_04\_FICHIER\_NON\_VIDE | Un fichier en entrée ne doit pas être vide | L'application cible doit contrôler que le fichier en entrée n'est pas vide | Technique |
| MOE | EX\_TEC\_05\_FICHIER\_TEXTE | Un fichier en entrée doit être un fichier texte | L'application cible doit contrôler que le fichier en entrée est un fichier texte (pas un mp3, un wav, …) | Technique |
| MOE | EX\_TEC\_06\_TRAITEMENT\_ENCODAGE\_UTF8 | Tous les fichiers gérés par l'application doivent être transcodés en interne en UTF-8. | Tous les fichiers gérés par l'application doivent être transcodés en interne en UTF-8. | Technique |
| MOE | EX\_TEC\_07\_CONTROLE\_ENCODAGE\_UTF8 | Contrôle de l'encodage UTF-8 des fichiers | L'application cible doit contrôler que les fichiers internes sont encodés en UTF-8 | Technique |
| MOE | EX\_TEC\_08\_DETECTION\_TYPE\_FICHIER | Détection du type de fichier (HIT, HISTO\_F07, DARWIN, …) | L'application cible doit détecter le type du fichier entrant (HIT, HISTO\_F07, DARWIN, …) et n'accepter en entrée que ces types de fichiers trafic | Technique |
| MOE | EX\_TEC\_09\_FICHIER\_SANS\_DOUBLONS\_LIGNE | Un fichier en entrée ne doit pas comporter de lignes en doublons | L'application cible doit détecter les lignes en doublon et éliminer les surplus | Technique |
| MOE | EX\_TEC\_10\_FICHIER\_SANS\_LIGNES\_VIDES | Un fichier en entrée ne doit pas comporter de lignes vides | L'application cible doit détecter les lignes vides et les éliminer | Technique |
| MOE | EX\_TEC\_11\_CHAMP\_RENSEIGNE | Un champ doit être renseigné dans une ligne | L'application doit détecter les champs non renseignés et leur appliquer une règle de gestion | Technique |
| MOE | EX\_TEC\_12\_FORMAT\_CHAMP | Un champ doit être au bon format dans une ligne | L'application doit détecter les champs au mauvais format et leur appliquer une règle de gestion | Technique |
| MOE | EX\_TEC\_13\_NOMENCLATURE\_CHAMP | Certains champs doivent respecter une nomenclature | L'application doit détecter les champs ne respectant pas une nomenclature et leur appliquer une règle de gestion | Technique |
| MOE | EX\_TEC\_14\_REGEX\_CHAMP | Certains champs doivent respecter une expression régulière (RegEx) | L'application doit détecter les champs ne respectant pas une expression régulière et leur appliquer une règle de gestion | Technique |
| MOE | EX\_TEC\_15\_CHAMPS\_REDHIBITOIRES | Un contrôle défavorable sur certains champs dits "rédhibitoires" doit aboutir au retrait de toute la ligne | L'application doit retirer toute la ligne du fichier si un de ses champs rédhibitoires est mal formé | Technique |
| MOE | EX\_TEC\_16\_ANNEE\_EXPLOITATION | L'année de toutes les sections d'un lot doit être l'année d'exploitation | L'application doit retirer toute la ligne du fichier si l'année de la ligne n'est pas l'année d'exploitation | Technique |
| MOE | EX\_TEC\_17\_LOCALISATIONS\_RENSEIGNEES | Les localisations de début et de fin de toutes les sections d'un lot doivent être renseignées | L'application doit retirer toute la ligne du fichier si une des localisations de la ligne n'est pas renseignée | Technique |
| MOE | EX\_TEC\_18\_SENS\_CONFONDUS | Les trafics sens confondus doivent être égaux à la somme des trafics sens croissants et décroissants pour une même section | L'application doit signaler les erreurs dans les cumuls de sens | Technique |
| MOE | EX\_TEC\_19\_CONTROLES\_ACCESSIBLES | Les contrôles doivent être accessibles pour les recetteurs | L'application doit mettre à disposition des recettes techniques et fonctionnelles un moyen d'accès aux contrôles | Technique |
| MOE | EX\_TEC\_20\_STOCKER\_RAPPORT\_CONTROLE | Les rapports de contrôles doivent être stockés sur le serveur | L'application doit permettre le stockage des rapports de contrôle sur le serveur pendant une période fixée par la MOA | Technique |
| MOE | EX\_TEC\_21\_EXPORTER\_RAPPORT\_CONTROLE | Les rapports de contrôle stockés sur le serveur doivent pouvoir être exportés vers les disques durs des acteurs | L'application doit permettre aux acteurs d'importer sur leur disque dur les rapports de contrôle stockés sur le serveur | Technique |
| MOE | EX\_TEC\_22\_ECRASER\_RAPPORTS\_CONTROLES | Les rapports de contrôle stockés sur le serveur doivent pouvoir être écrasés automatiquement après un délai fixé par la MOA ou manuellement | L'application doit pouvoir écraser automatiquement les rapports de contrôle sur le serveur après un délai fixé par la MOA. Elle doit en outre permettre à un Administrateur applicatif d'écraser manuellement les rapports de contrôle | Technique |
| MOE | EX\_TEC\_23\_AGREGER\_RAPPORTS\_CONTROLES | Les rapports de contrôle doivent pouvoir être agrégés afin que la MOA dispose d'un rapport unique pour une période déterminée | L'application doit pouvoir servir des rapports de contrôle concaténables afin d'obtenir un rapport unique pour une période déterminée | Technique |
| MOE | EX\_TEC\_24\_RAPPORTS\_CONTROLE\_CSV | Les rapports de contrôle doivent être enregistrés au format csv | L'application doit pouvoir servir des rapports de contrôle au format csv | Technique |
| MOE | EX\_TEC\_25\_RAPPORTS\_CONTROLE\_TXT | Les rapports de contrôle doivent être enregistrés au format textuel | L'application doit pouvoir servir des rapports de contrôle au format textuel | Technique |
| MOE | EX\_TEC\_26\_MEMORISER\_NUMEROS\_ORIGINAUX | Les rapports de contrôle doivent utiliser les numéros de ligne originaux (numéros de ligne dans le fichier soumis) | L'application doit servir des rapports de contrôle utilisant les numéros de ligne originaux (numéros de ligne dans le fichier soumis) | Technique |

### Synthèse des RG capturées

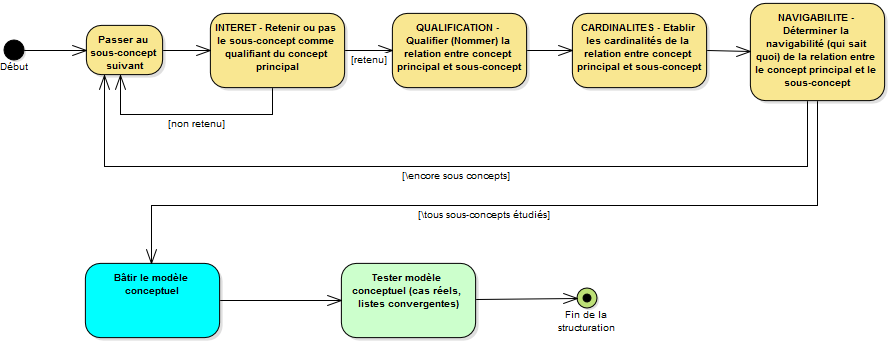


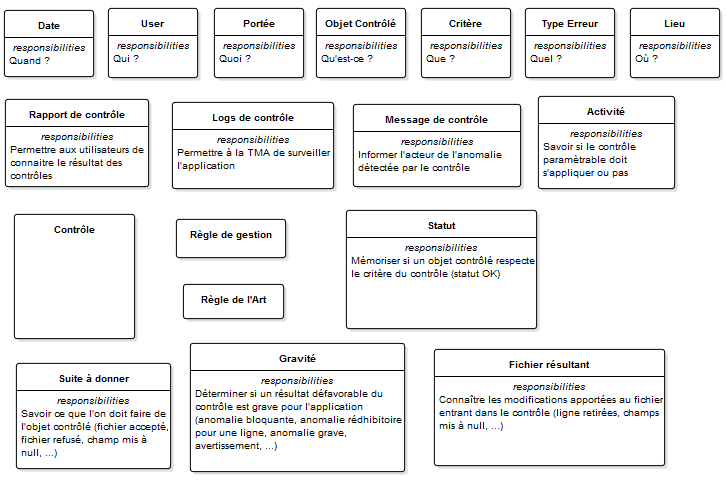
| **Dem.** | **IDENTIFIANT** | **DESCRIPTION** | **PORTEE** | **EXIGENCE ASSOCIEE** |
| --- | --- | --- | --- | --- |
| MOA | RG-01\_FICHIER\_CONTROLE\_VALIDITE | L'application cible doit contrôler la validité des fichiers acceptés en entrée | Tous Fichiers | EX\_FONCT\_01\_CONTROLE\_DE\_VALIDITE |
| MOE | RG-100\_TRUE\_SI\_CONTROLE\_FAVORABLE | 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 doublons. | Tous Fichiers |  |
| MOA | RG-101\_TOUT\_CONTROLE\_RAPPORTE | L'application cible doit fournir des rapports de contrôle expliquant le résultat de tous ses contrôles et traitements | Tous Fichiers | EX\_FONCT\_03\_RAPPORT\_CONTROLE |
| MOA | RG-102\_REGLES\_ART | L'application cible doit exécuter tous les contrôles conformes aux règles de l'art | Tous Fichiers | EX\_FONCT\_06\_REGLE\_ART |
| MOA | RG-02\_FICHIER\_CONTROLE\_METIER | L'application cible doit contrôler que les données métier satisfont à des règles de gestion | Tous Fichiers | EX\_FONCT\_02\_CONTROLE\_METIER |
| MOA | RG-01-01\_FICHIERS\_TEXTE | L'application cible ne doit admettre en entrée que des fichiers texte (HIT, HISTO\_F07, DARWIN, ,,,) | Tous Fichiers | EX\_TEC\_05\_FICHIER\_TEXTE |
| MOE | RG-01-02\_TRANSCODAGE\_UTF-8 | Tous les fichiers gérés par l'application doivent être transcodés en interne en UTF-8. | Tous Fichiers | EX\_TEC\_06\_TRAITEMENT\_ENCODAGE\_UTF8 |
| MOE | RG-01-03\_FICHIERS\_ENCODAGE\_UTF-8 | L'application cible doit contrôler que les fichiers internes sont encodés en UTF-8 | Tous Fichiers | EX\_TEC\_07\_CONTROLE\_ENCODAGE\_UTF8 |
| MOA | RG-01-04\_FICHIERS\_TYPE\_TRAFIC | L'application cible doit détecter le type du fichier entrant (HIT, HISTO\_F07, DARWIN, …) et n'accepter en entrée que ces types de fichiers trafic | Tous Fichiers | EX\_TEC\_08\_DETECTION\_TYPE\_FICHIER |
| MOE | RG-01-05\_ELIMINATION\_DOUBLONS\_INTERLIGNES | Aucun fichier ne doit comporter de lignes en doublons. Si des doublons (triplettes, …) sont trouvés, seul le premier exemplaire doit être conservé. Les autres sont éliminés et le programme continue. | Tous Fichiers | EX\_TEC\_09\_FICHIER\_SANS\_DOUBLONS\_LIGNE |
| MOE | RG-01-06\_ELIMINATION\_LIGNES\_VIDES | Aucun fichier ne doit comporter de lignes vides. Si des lignes vides sont trouvées, elles sont éliminées eet le programme continue. | Tous Fichiers | EX\_TEC\_10\_FICHIER\_SANS\_LIGNES\_VIDES |
| MOA | RG-01-07\_ELIMINATION\_LIGNE\_REDHIBITOIRE | Aucun fichier ne doit comporter de lignes avec des champs "redhibitoires" mal formés. Si une telle ligne est trouvée, elle est éliminée et le programme continue | Tous Fichiers | EX\_TEC\_15\_CHAMPS\_REDHIBITOIRES |
| MOA | RG\_DETECTION\_HIT\_01\_LIGNES | Dans un HIT, toutes les lignes doivent comporter exactement 520 caractères | HIT | EX\_TEC\_08\_DETECTION\_TYPE\_FICHIER |
| MOA | RG-200-1\_CONTROLES PARAMETRABLES | Un contrôle ne doit s'appliquer et rapporter que si il est activé | Tous Fichiers | EX\_FONCT\_07\_CONTROLES\_PARAMETRABLES |
| MOA | RG-300-1\_ANNEE\_EXPLOITATION | L'application doit retirer toute la ligne du fichier si l'année de la ligne n'est pas l'année d'exploitation. Si une telle ligne est trouvée, elle est éliminée et le programme continue | Tous Fichiers | EX\_TEC\_16\_ANNEE\_EXPLOITATION |
| MOA | RG-300-2\_LOCALISATIONS\_RENSEIGNEES | L'application doit retirer toute la ligne du fichier si une des localisations de la ligne n'est pas renseignée, Si une telle ligne est trouvée, elle est éliminée et le programme continue. | Tous Fichiers | EX\_TEC\_17\_LOCALISATIONS\_RENSEIGNEES |
| MOA | RG-300-3\_SENS\_CONFONDUS | Les sens confondus doivent être égaux à la somme des sens croissants et décroissants pour une même section. Si une telle ligne est trouvée, l'application rédige un message d'alerte, la ligne est conservée et le programme continue | Tous Fichiers | EX\_TEC\_18\_SENS\_CONFONDUS |

## Affiner et Structurer les concepts

Objectif :

* Décider de l'**intérêt** des sous-concepts capturés pour qualifier le concept Contrôle (sous-concepts retenus ou pas dans le modèle conceptuel)
* **Qualifier** les relations entre concepts
* Révéler les **cardinalités** des relations entre concepts
* Révéler la **navigabilité** des relations entre concepts
* Bâtir le **modèle conceptuel**
* **Tester** le modèle conceptuel (cas réels, listes convergentes pour vérifier les dépendances)





### Concept Date

Le concept Date permet de savoir quand a eu lieu le contrôle.

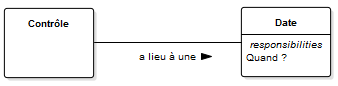
#### Intérêt du concept Date

Ce concept est indispensable pour identifier un contrôle. La date du contrôle doit en outre apparaître dans les rapports de contrôle.

Le concept de date du contrôle est donc **retenu** pour qualifier un contrôle.

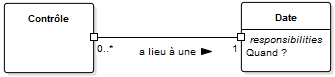
#### Qualification de la relation entre Contrôle et Date

Un Contrôle **a lieuà une** Date de contrôle.



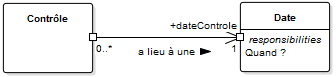
#### Cardinalités entre Contrôle et Date

* Un contrôle a lieu à **une et une seule** Date de contrôle donnée.
* A une Date donnée peuvent s'exécuter **plusieurs contrôles ou aucun**.



#### Navigabilité entre Contrôle et Date

* Un Contrôle **sait** toujours à quelle Date il a eu lieu
* Une Date ne **sait pas** quels contrôles ont été exécutés à cette Date.



### Concept User

Le User désigne traditionnellement l'acteur qui a déclenché une action en informatique. Dans une application web, c'est l'internaute loggé en session web qui est en train de réaliser un cas d'utilisation. Il a le plus souvent un nom, un prénom, un Login, un mot de passe, un profil applicatif (profil Cerbere au Ministère), …

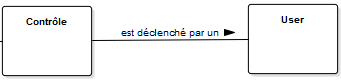
#### Intérêt du concept User

Un contrôle n'a pas besoin de toutes ces informations. En revanche, il est nécessaire de connaître le **nom** du User qui a déclenché un contrôle à des fins de tri. Par exemple, il est nécessaire de savoir que c'est la DIRA qui vient de déposer le fichier HIT\_DIRA\_2014, et que ce dépôt a déclenché un contrôle prouvant qu'un département était mal formé.

User est donc un concept **retenu** même si c'est *le nom du User qui sera utilisé*.

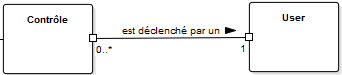
#### Qualification de la relation entre Contrôle et User

Un Contrôle **est déclenché par un** User



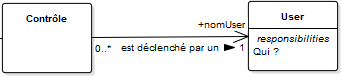
#### Cardinalités entre Contrôle et User

* Un Contrôle est déclenché par **un et un seul** User.
* Un User peut déclencher **plusieurs contrôles (ou zéro)**.



#### Navigabilité entre Contrôle et User

* Un Contrôle **sait** toujours quel User l'a déclenché
* Un User ne **sait pas** quels Contrôles il a déclenché dans l'application TraficWeb. On obtiendra cette information en triant sur le User le rapport de contrôle sous forme tabulaire (csv).



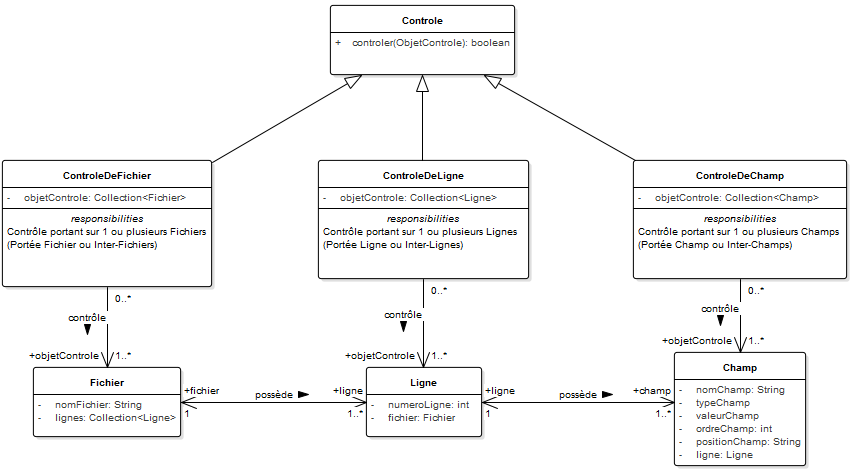
### Concept Portée

Le concept Portée détermine sur quoi (un fichier, inter-fichiers, une ligne, inter-lignes, un champ, inter-champs) porte un contrôle.

Ce concept est intéressant pour typer les messages de contrôle qui diffèrent en fonction de l'objet contrôlé.

* Par exemple, un contrôle inter-fichiers qui contrôlera que "le fichier A est plus grand que le fichier B" nécessitera un message du type : "true, le fichier A est plus grand que B".
* Un contrôle de portée fichier sur le contenu du fichier HIT\_DIRA\_2014 nécessitera un message du type : "le fichier HIT\_DIRA\_2014 n'est pas vide. OK".
* Un contrôle sur le renseignement du champ Département d'une ligne n dans un fichier HIT\_DIRA\_2014 nécessitera un message du type : "le champ Département (1er champ) situé entre les positions 0 et 3 inclus à la ligne n du fichier HIT\_DIRA\_2014 n'est pas renseigné".

Le concept Portée est donc intéressant pour qualifier l'objet contrôlé et le message de contrôle à générer. Ce concept peut aider le Développeur à packager son application en rangeant les contrôles de portée inter-fichiers dans un package, les contrôles de portée fichier dans un autre, puis les contrôles de portée inter-lignes à part, etc…



En revanche, il n'y a pas d'intérêt pour la MOA à trier les rapports de contrôles sur la portée. Il n'y a donc pas lieu de qualifier les contrôles avec ce concept Portée. Le code s'en chargera.

Le concept Portée n'est donc **pas retenu** pour qualifier le concept Contrôle. Il est contenu dans le **type** du contrôle

### Concept Objet Contrôlé

Tout contrôle s'applique à un "objet contrôlé" (2 ou plusieurs fichiers, un seul fichier, deux ou plusieurs lignes, une seule ligne, deux ou plusieurs champs, un seul champ).

L'objet contrôlé est par exemple : "les fichiers HIT\_DIRA\_2013 et HIT\_DIRA\_2014", ou "le fichier HIT\_DIRA\_2014", ou encore "le champ Département de la ligne n du fichier HIT\_DIRA\_2014", …

Par exemple :

* Un contrôle déterminera si le fichier HIT\_DIRA\_2014 n'est pas vide
* Un autre contrôle déterminera si la 3ème ligne du fichier HIT\_DIRA\_2014 n'est pas vide.
* Un contrôle déterminera si le champ Département d'une ligne n d'un fichier HIT\_DIRA\_2014 est bien renseigné.
* Un autre contrôle déterminera si le champ Département d'une ligne n d'un fichier HIT\_DIRA\_2014 est bien numérique.
* Un autre contrôle déterminera si l'écart entre le Trafic Moyen Journalier Annuel (TMJA) de l'année n et la TMJA de l'année n-1 respecte une certaine fourchette dans une ligne n du fichier HIT\_DIRA\_2014.

La MOA aura besoin de comptabiliser le nombre de champs Département mal renseignés dans le fichier HIT\_DIRA\_2014 par exemple.

Le concept Objet Contrôlé est donc à priori retenu.

Mais l'exigence fonctionnelle EX\_FONCT\_07\_CONTROLES\_PARAMETRABLES précise que *chaque contrôle doit pouvoir être activé/désactivé individuellement*.

Le développeur devra donc coder (pour le champ Département par exemple) :

* Un contrôle *ControleChampDepartementRenseigne* qui contrôlera que le Département est bien renseigné à chaque ligne d'un fichier.
* Un contrôle *ControleChampDepartementNumerique* qui contrôlera que le Département est bien homogène à un numérique à chaque ligne d'un fichier.
* Un contrôle *ControleChampDepartementDansNomenclature* qui contrôlera que le Département respecte une nomenclature à chaque ligne du fichier
* Un contrôle *ControleChampDepartementRegex* qui contrôlera que le Département respecte une expression régulière (RegEx) à chaque ligne du fichier.

On constate donc que chaque contrôle **porte** entre autres dans son **type** l'objet contrôlé (Département ci-dessus). Comme pour le concept Portée, le code se charge de qualifier le contrôle concernant l'objet contrôlé.

Le concept Objet Contrôlé n'est donc **pas retenu** pour qualifier le concept Contrôle. Il est contenu dans le **type** du contrôle.

### Concept Critère

Le concept de Critère détermine *ce que le contrôle a contrôlé*.

Par exemple :

|  |  |
| --- | --- |
| Contrôle | Critère |
| *ControleChampDepartementRenseigne* | Le champ Département est renseigné |
| *ControleChampDepartementNumerique* | Le champ Département est homogène à un numérique |
| *ControleChampDepartementDansNomenclature* | Le champ Département respecte la nomenclature du Département dans un HIT |
| *ControleChampDepartementRegex* | Le champ Département respecte l'expression régulière (RegEx) imposée dans les HIT |

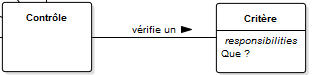
#### Intérêt du concept Critère

Ce concept Critère est intéressant pour pouvoir trier les rapports de contrôles en fonction des critères. Il qualifie le contrôle dans la mesure où il précise ce qui a été contrôlé.

Le concept Critère est **retenu**.

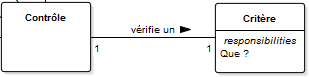
#### Qualification de la relation entre Contrôle et Critère

Un Contrôle **vérifie un** Critère



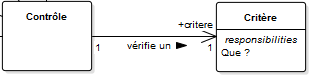
#### Cardinalités entre Contrôle et Critère

* Un Contrôle vérifie **un et un seul** Critère
* Un Critère est vérifié par **un et un seul** Contrôle.



#### Navigabilité entre Contrôle et Critère

* Un contrôle **sait** toujours quel Critère il vérifie.
* Un Critère ne **sait pas** quel Contrôle le vérifie. On obtiendra cette information en triant sur le Critère le rapport de contrôle sous forme tabulaire (csv).



### Concept Type d'Erreur

Le concept Type d'Erreur (TypeErreur) détermine quel *type d'anomalie* le contrôle a détecté.

Par exemple :

| **Type d'Erreur** | **Contrôle** | **Message de contrôle** |
| --- | --- | --- |
| Non renseigné | Controle  ChampDepartementRenseigne | Le Département à la ligne n du fichier HIT\_DIRA\_2014 n'est pas renseigné |
| Non renseigné | Controle  ChampTMJARenseigne | La TMJA de l'année N à la ligne n du fichier HIT\_DIRA\_2014 n'est pas renseignée |
| Mauvais format | Controle  ChampDepartementNumerique | Le Département à la ligne n du fichier HIT\_DIRA\_2014 n'est pas homogène à un entier |

#### Intérêt du concept TypeErreur

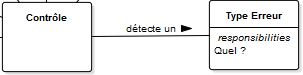
La MOA aura besoin de trier les rapports de contrôle pour comptabiliser les anomalies en fonction du TypeErreur.

Par exemple, comptabiliser le nombre de champs "non renseigné" dans un certain fichier.

Le concept TypeErreur est **retenu**.

#### Qualification de la relation entre Contrôle et TypeErreur

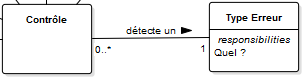
Un contrôle **détecte un** TypeErreur.



#### Cardinalités entre Contrôle et TypeErreur

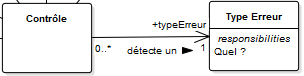
Du fait de l'exigence fonctionnelle EX\_FONCT\_07\_CONTROLES\_PARAMETRABLES qui impose une classe concrète de contrôle pour chaque type d'erreur afin de pouvoir activer/desactiver individuellement chaque contrôle :

* Un Contrôle détecte **un et un seul** TypeErreur
* Un TypeErreur peut être détecté par **plusieurs contrôles (ou zéro)**. C'est le cas par exemple du type d'erreur "non renseigné" qui peut être détecté pour tous les champs d'un fichier (TMJA, Département, …) et donc par divers contrôles individualisés *ControleChampDepartementRenseigne*, *ControleChampTMJARenseigne*, …).



#### Navigabilité entre Contrôle et TypeErreur

* Un Contrôle **sait** toujours quel TypeErreur il détecte
* Un TypeErreur ne **sait pas** quel Contrôle le détecte. On obtiendra cette information en triant sur le TypeErreur le rapport de contrôle sous forme tabulaire (csv).



### Concept Lieu

Le concept de Lieu détermine où le contrôle a eu lieu dans l'application (contrôle de surface qui a lieu dès la lecture d'un fichier déposé, contrôle métier qui a lieu dans la couche métier après que le fichier déposé ait été décomposé, …)

Ce concept peut aider le Développeur à packager son application en rangeant les contrôles de surface dans un package, les contrôles métier dans un autre, etc…

Mais la MOA n'aura aucun intérêt à trier les rapports de contrôle en fonction des contrôles de surface, des contrôle métier, …

Le concept de Lieu n'est donc **pas retenu**.

### Concept Activité

Le concept Activité détermine si un contrôle doit être exécuté ou pas. Il est lié à la notion de contrôle paramétrable.

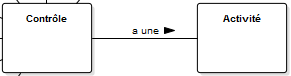
#### Intérêt du concept Activité

La MOA aura besoin de paramétrer les contrôles afin de les activer/désactiver. Elle aura également besoin d'obtenir de l'application la liste des contrôles implémentés et de savoir lesquels sont actifs.

Le concept Activité est donc **retenu**.

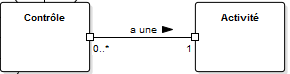
#### Qualification de la relation entre Contrôle et Activité

Un Contrôle a une Activité (il est actif ou inactif)



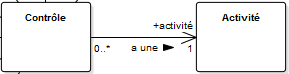
#### Cardinalités entre Contrôle et Activité

* Un Contrôle a une et une seule Activité (il est actif ou inactif).
* Une Activité peut concerner 0 ou plusieurs Contrôles.



#### Navigabilité entre Contrôle et Activité

* Un Contrôle **sait** toujours s'il est actif ou inactif.
* Une Activité **ne sait pas** quels contrôles sont actifs (ou inactifs).



### Concept Statut

Le concept Statut sert à mémoriser si un contrôle a matché favorablement (l'objet contrôlé respecte le critère de contrôle et le statut est OK) ou défavorablement (l'objet contrôlé ne respecte pas le critère de contrôle et le statut est KO).

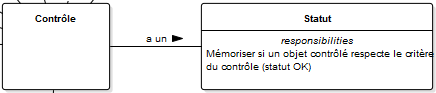
#### Intérêt du concept Statut

La MOA aura besoin de trier les rapports de contrôle pour savoir quels sont tous les contrôles qui ont matché défavorablement.

Le concept Statut est donc **retenu**.

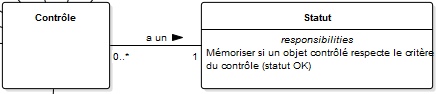
#### Qualification de la relation entre Contrôle et Statut

Un Contrôle a un Statut après son exécution.



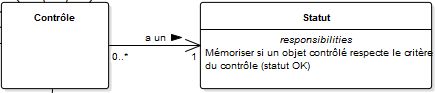
#### Cardinalités entre Contrôle et Statut

* Un Contrôle a un et un seul Statut.
* Un Statut (OK/KO) peut concerner 0 ou plusieurs Contrôles.



#### Navigabilité entre Contrôle et Statut

* Un Contrôle sait toujours quel est son Statut (OK/KO) après son exécution.
* Un Statut ne sait pas quels contrôles sont concernés (par exemple, le Statut KO ne sait pas quels contrôles sont KO). Cette information sera accessible en triant les rapports de contrôle sur le statut.



### Concept Gravité

Le concept Gravité permet de mémoriser si un Contrôle a un impact important pour l'application (anomalie bloquante pour un fichier, anomalie rédhibitoire pour une ligne, avertissement, …). Par exemple, un Département non renseigné ou mal formé dans une ligne d'un fichier HIT est une anomalie rédhibitoire. En effet, cette ligne ne pourra pas être convenablement localisée et perd tout intérêt.

#### Intérêt du concept Gravité

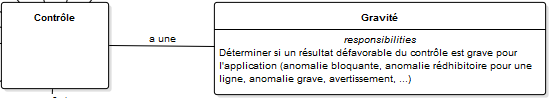
La MOA aura besoin de trier les rapports de contrôle pour connaître toutes les anomalies rédhibitoires pour des lignes de fichier par exemple.

Le concept Gravité est donc **retenu**.

#### Qualification de la relation entre Contrôle et Gravité

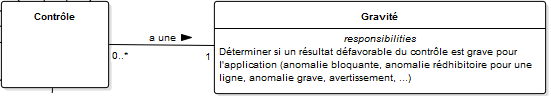
Un Contrôle **a une** Gravité.

Par exemple, un contrôle sur le format numérique d'un Département dans une ligne d'un fichier HIT peut déclencher une anomalie rédhibitoire pour la ligne.



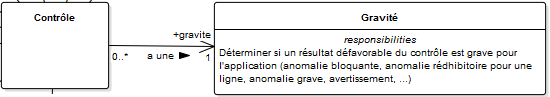
#### Cardinalités entre Contrôle et Gravité

* Un Contrôle a **une et une seule** Gravité.
* Une Gravité peut concerner **0 ou plusieurs** Contrôles.



#### Navigabilité entre Contrôle et Gravité

* Un Contrôle **sait** toujours quelle est sa Gravité.
* Une Gravité ne **sait pas** quels Contrôles elle concerne.



### Concept Suite A Donner

Le concept Suite A Donner permet de savoir ce que le contrôle doit faire avec l'objet contrôlé *en cas de contrôle défavorable*.

Par exemple, un contrôle de surface défavorable de format numérique d'un champ Département dans une ligne d'un fichier HIT devra supprimer la ligne.

Un contrôle métier défavorable de respect d'une fourchette sur le **T**rafic **M**oyen **J**ournalier **A**nnuel (TMJA) dans une ligne de fichier HIT devra annuler cette TMJA (si **R**ègle de **G**estion RG) mais conserver la ligne.

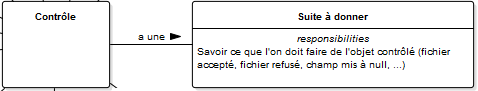
#### Intérêt du Concept Suite A Donner

La MOA aura besoin de connaître les lignes supprimées, les champs annulés, les champs conservés mais ayant déclenché un avertissement, …

Le concept Suite A Donner est **retenu**.

#### Qualification de la relation entre Contrôle et Suite A Donner

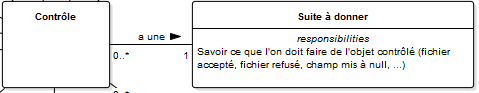
Un Contrôle **a une** Suite A Donner.



#### Cardinalités entre Contrôle et Suite A Donner

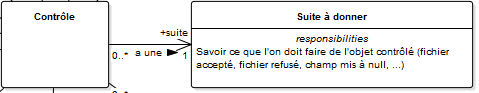
Un Contrôle a **une et une seule** Suite A Donner à un objet contrôlé.

Une Suite A Donner concerne **0 ou plusieurs** Contrôles.



#### Navigabilité entre Contrôle et Suite A Donner

* Un Contrôle **sait** toujours quelle Suite il doit donner à l'objet contrôlé.
* Une Suite A Donner ne **sait pas** quels Contrôles sont concernés.



### Concept Fichier Résultant

Le concept Fichier Résultant permet de mémoriser les changements apportés à un fichier par un contrôle. Certains contrôles peuvent en effet transformer le fichier entrant en lui retirant des lignes, en annulant des champs, …

Ce concept est primordial pour pouvoir enchaîner les contrôles

#### Intérêt du Concept Fichier Résultant

La MOA aura besoin de connaître le fichier résultant de la **chaîne des contrôles** de l'application.

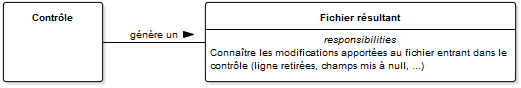
En outre, les développeurs ont impérativement besoin d'accéder au Fichier Résultant d'un contrôle puisque le contrôle suivant ne doit s'appliquer qu'au Fichier Résultant.

Le concept Fichier Résultant est **retenu**.

#### Qualification de la relation entre Contrôle et Fichier Résultant

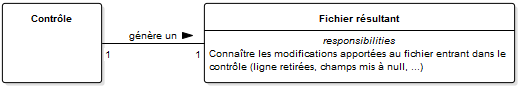
Un Contrôle **génère un** Fichier Résultant.

Ce Fichier Résultant peut le cas échéant être le même que le fichier en entrée (cas des contrôles qui n'exécutent aucun traitement).



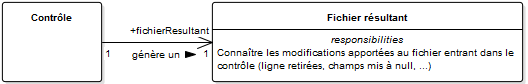
#### Cardinalités entre Contrôle et Fichier Résultant

* Un Contrôle génère **un et un seul** Fichier Résultant.
* Un Fichier Résultant et généré par **un et un seul** Contrôle.



#### Navigabilité entre Contrôle et Fichier Résultant

* Un Contrôle **sait** toujours quel Fichier Résultant il a généré.
* Un Fichier Résultant ne **sait pas** quel Contrôle l'a généré.



### Concept Règle de Gestion

Un contrôle met en application 0, 1 ou plusieurs Règles de Gestion.

#### Intérêt du Concept Règle de Gestion

La MOA n'a pas besoin de trier les rapports de contrôle en fonction des règles de gestion.

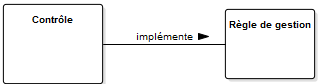
En revanche, le **chef de projet informatique** et le **développeur (TMA)** peuvent souhaiter vérifier que toutes les règles de gestion ont bien été implémentées lors de l'administration des contrôles.

Il est intéressant que le code "sache dire " où et comment les règles de gestion ont été implémentées dans le cadre de l'élaboration du projet informatique (traçabilité des concepts dans un projet Entreprise Architect par exemple).

Le concept de Règle de Gestion est donc **retenu**.

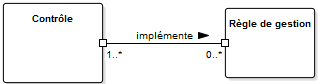
#### Qualification de la relation entre Contrôle et Règle de Gestion

Un Contrôle **implémente** des Règles de Gestion



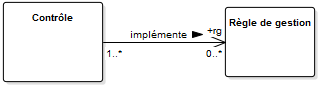
#### Cardinalités entre Contrôle et Règle de Gestion

* Un Contrôle implémente **0, 1 ou plusieurs** Règles de Gestion.
* Un Règle de Gestion est implémentée par **1 ou plusieurs** Contrôles.



#### Navigabilité entre Contrôle et Règle de Gestion

* Un Contrôle **sait** quelles Règles de Gestion il implémente.
* Une Règle de Gestion ne **sait pas** Quels Contrôles l'implémentent. Nous aurons accès à cette information en triant le rapport d'administration des contrôles – qui liste pour chaque contrôle les règles de gestion qu'il implémente - sur les règles de gestion.



### Concept Règle de l'Art

Un contrôle met en application 0, 1 ou plusieurs Règles de l'Art.

#### Intérêt du Concept Règle de l'Art

La MOA n'a pas besoin de trier les rapports de contrôle en fonction des règles de l'art.

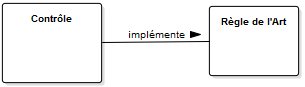
En revanche, le **chef de projet informatique** et le **développeur (TMA)** peuvent souhaiter vérifier que toutes les règles de l'art ont bien été implémentées lors de l'administration des contrôles.

Il est intéressant que le code "sache dire " où et comment les règles de l'art ont été implémentées dans le cadre de l'élaboration du projet informatique (traçabilité des concepts dans un projet Entreprise Architect par exemple).

Le concept de Règle de l'art est donc **retenu**.

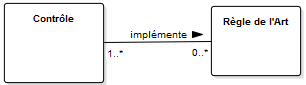
#### Qualification de la relation entre Contrôle et Règle de l'Art

Un Contrôle **implémente** des Règles de l'Art



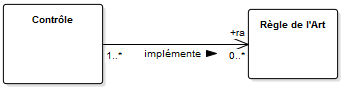
#### Cardinalités entre Contrôle et Règle de l'Art

* Un Contrôle implémente **0, 1 ou plusieurs** Règles de l'Art.
* Un Règle de l'Art est implémentée par **1 ou plusieurs** Contrôles.



#### Navigabilité entre Contrôle et Règle de l'Art

* Un Contrôle **sait** quelles Règles de l'Art il implémente.
* Une Règle de l'Art ne **sait pas** Quels Contrôles l'implémentent. Nous aurons accès à cette information en triant le rapport d'administration des contrôles – qui liste pour chaque contrôle les règles de l'art qu'il implémente - sur les règles de l'art.



### Concept Rapport de Contrôle

Le concept Rapport de Contrôle détermine ce dont l'acteur va bénéficier pour prendre connaissance de la qualité d'un fichier de trafic soumis à l'application.

Le Rapport de contrôle doit être au format csv pour être exportable vers des tableurs et au format textuel pour des **I**nterfaces **H**omme-**M**achine (IHM) autonomes à fenêtres (Desktop).

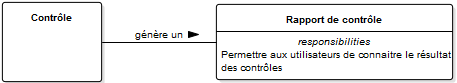
#### Intérêt du Concept Rapport de Contrôle

Tous les acteurs auront besoin de consulter les rapports de contrôle.

Le concept Rapport de Contrôle est donc **retenu**.

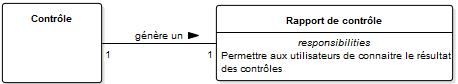
#### Qualification de la relation entre Contrôle et Rapport de Contrôle

Un Contrôle **génère un** Rapport de Contrôle.



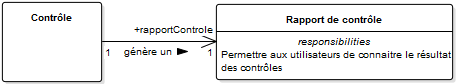
#### Cardinalités entre Contrôle et Rapport de Contrôle

* Un Contrôle génère **un et un seul** Rapport de Contrôle.
* Un Rapport de Contrôle est généré par **un et un seul** Contrôle.



#### Navigabilité entre Contrôle et Rapport de Contrôle

* Un Contrôle **sait** quel Rapport de Contrôle il a généré.
* Un Rapport de Contrôle ne **sait pas** quel Contrôle l'a généré.



### Concept Log de Contrôle

Le concept Log de contrôle détermine ce que les acteurs applicatifs (Administrateur Applicatif, Développeur (TMA), Recetteur Technique, Centre Serveur) vont pouvoir observer à postériori afin d'assurer le suivi de l'application informatique.

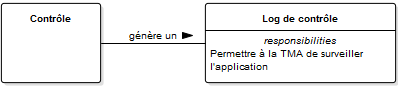
#### Intérêt du Concept Log de Contrôle

L'**Administrateur Applicatif**, le **Développeur (TMA)**, le **Recetteur Technique** et le **Centre Serveur** peuvent avoir besoin de consulter les Logs de Contrôle.

Le concept Log de Contrôle est **retenu**.

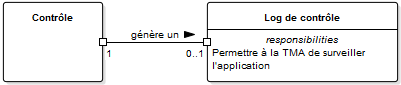
#### Qualification de la relation entre Contrôle et Log de Contrôle

Un Contrôle **génère un** Log de Contrôle.



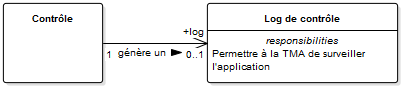
#### Cardinalités entre Contrôle et Log de Contrôle

* Un Contrôle génère **zéro ou un seul** Log de Contrôle.
* Un Log de Contrôle est généré par **un et un seul** Contrôle.



#### Navigabilité entre Contrôle et Log de Contrôle

* Un Contrôle **sait** quel Log de Contrôle il a généré.
* Un Log de Contrôle ne **sait pas** quel Contrôle l'a généré.



### Concept Message de Contrôle

Le concept Message de Contrôle détermine ce que l'application "dit" lorsqu' un contrôle a matché favorablement ou défavorablement.

Par exemple, un contrôle de format numérique sur le champ Département dans une ligne d'un fichier HIT pourra émettre le message : "le Département de la ligne 3 du fichier HIT\_DIRA\_2014 n'est pas homogène à un entier : '20A'. Cette ligne 3 est donc rejetée".

Le Message de Contrôle est encapsulé dans un Rapport de Contrôle ou un Log. Ce n'est donc pas directement un concept de Contrôle. C'est un concept lié à Rapport de Contrôle et Log de Contrôle.

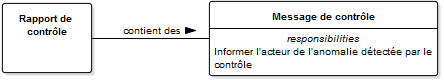
#### Intérêt du Concept Message de Contrôle

Tous les acteurs auront besoin de lire les messages de contrôle via le rapport de contrôle ou le Log.

Le concept Message de Contrôle est donc **retenu**.

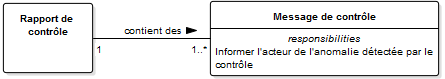
#### Qualification de la relation entre Rapport de Contrôle et Message de Contrôle

Un Rapport de Contrôle (ou Log) **contient** **des** Messages de Contrôle.



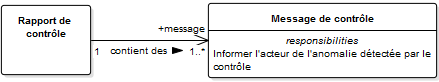
#### Cardinalités entre Rapport de Contrôle et Message de Contrôle

* Un Rapport de Contrôle contient **un ou plusieurs** Message de Contrôle.
* Un Message de Contrôle est contenu dans **un et un seul** Rapport de Contrôle.



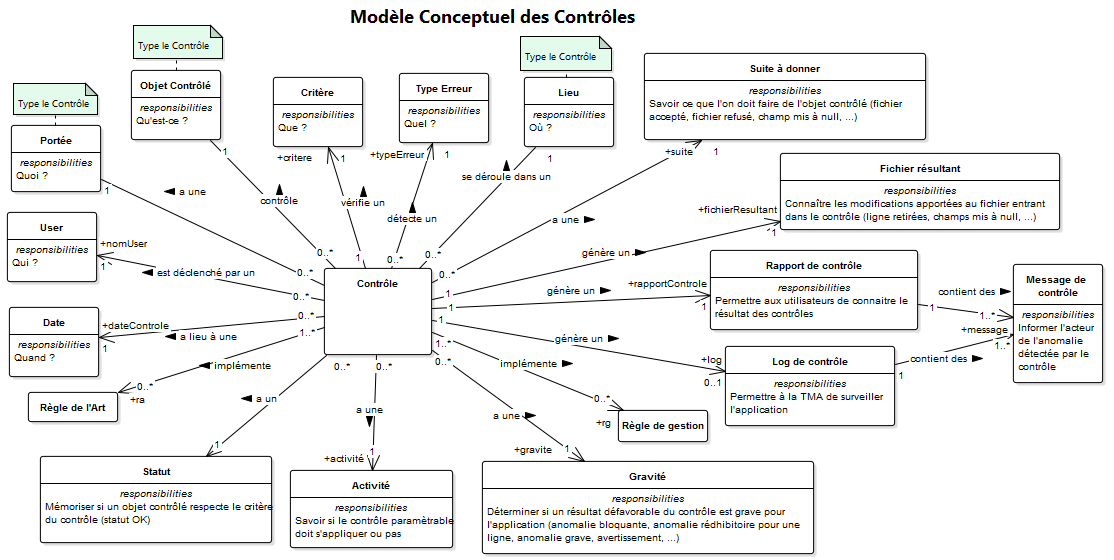
#### Navigabilité entre Rapport de Contrôle et Message de Contrôle

* Un Rapport de Contrôle **sait** quels Messages de Contrôle il contient.
* Un Message de Contrôle **ne sait pas** quel Rapport de Contrôle le contient.



## Bâtir le modèle conceptuel des Contrôles

Objectif : synthétiser le modèle conceptuel des Contrôles. A ce stade, nous avons bâti le modèle conceptuel suivant :



Il faut ensuite affiner ce modèle conceptuel en capturant les attributs (ce que doit "savoir" l'objet) et les méthodes (ce que doit "savoir faire" l'objet) de chaque concept.

### Capture des attributs et méthodes des concepts

Objectif : l'objectif de cette phase "capture des attributs et méthodes des concepts" est d'obtenir une première ébauche du modèle de classe. Les concepts sont progressivement transformés en classe (au sens Objet) et voient apparaître leurs attributs et leurs premières méthodes.

#### Attribut simple dateControle

Un Contrôle sait toujours à quelle Date il a eu lieu. La Date ne sait rien du Contrôle. Il est donc inutile de créer un Objet particulier (pure fabrication) pour la Date puisque notre concept Date ne doit rien faire de particulier. On peut utiliser directement la Date fournie par Java.

En pratique, il suffira de renseigner cet attribut avec la date système lors de l'exécution du contrôle.

#### Attribut simple userName

Un Contrôle sait toujours quel User l'a déclenché. Un User est toujours un objet qui connaît son nom, son prénom, son profil Cerbère, …

Mais le concept Contrôle n'a besoin que du nom du User, ou plus exactement du nom de l'organisation du User qui a déclenché le contrôle (DIRA, DIRE, DARWIN, …).

Il est donc inutile de mettre un attribut de type User puisque l'on ne se servira jamais de toutes ses informations lors des contrôles.

On peut se contenter d'un attribut simple userName et utiliser le type String fourni par Java puisque l'on utilisera une simple chaîne de caractères.

#### Attribut simple critere

Un Contrôle sait toujours quel critère il vérifie comme par exemple "le fichier soumis n'est pas vide" ou "le champ Département dans la ligne n du fichier HIT est renseigné".

Le critère est une simple chaine de caractères (String).

#### Attribut simple typeErreur (attribut à Label)

Un Contrôle sait toujours quel type d'erreur il vérifie (par exemple "nullité d'un fichier", "fichier vide", "ligne vide", "champ non renseigné", "champ de mauvais format", …).

Il est toujours plus facile de trier dans un tableau ce type de champs lorsqu'ils sont entiers numériques. En revanche, un label (String) est toujours plus lisible pour l'acteur. En outre, la MOA pourra le cas échéant modifier à postériori modifier le label à sa convenance.

Par exemple :

| **typeErreur** | **Label du typeErreur** |
| --- | --- |
| 1 | CHAMP NON RENSEIGNE |
| 2 | MAUVAIS FORMAT DE CHAMP |
| 3 | CHAMP HORS NOMENCLATURE |
| 4 | LIGNE VIDE |
| 5 | LIGNE SANS LOCALISATIONS |

L'attribut typeErreur est de type Java Integer.

L'attribut labelTypeErreur (attribut *dépendant* de typeErreur) est un String.

#### Attribut simple actif

Un contrôle sait toujours s'il est actif/inactif pour être exécuté ou pas. Le concept Activité ne sait rien du Contrôle.

Un type simple Java de type primitif boolean est donc suffisant.

#### Attribut simple statut (attribut à Label)

Un contrôle sait toujours si le contrôle qu'il vient d'effectuer a matché favorablement ou défavorablement. Le concept Statut ignore tout du Contrôle qui l'a généré.

Un type simple Java de type primitif boolean est donc suffisant.

Il est toujours plus facile de trier dans un tableau ce type de champs lorsqu'ils sont entiers numériques (1 ou 0) ou booléens (true, false).

En revanche, un label (String) est toujours plus lisible pour l'acteur. En outre, la MOA pourra le cas échéant modifier à postériori modifier le label à sa convenance.

L'usage est de dire qu'un contrôle est OK s'il a matché favorablement et KO s'il a matché défavorablement.

L'attribut statut est de type Java primitif boolean.

L'attribut labelStatut (attribut *dépendant* de statut) est un String.

| **statut** | **labelStatut** |
| --- | --- |
| true | OK |
| false | KO |

#### Attribut simple gravite (attribut à Label)

Un contrôle connaît toujours sa gravité et le concept Gravité ne sait rien des concepts Contrôle qu'elle concerne.

La gravité et donc un attribut simple (pas un objet).

Il est toujours plus facile de trier dans un tableau ce type de champs lorsqu'ils sont entiers numériques. En revanche, un label (String) est toujours plus lisible pour l'acteur. En outre, la MOA pourra le cas échéant modifier à postériori modifier le label à sa convenance.

Par exemple :

| **gravite** | **labelGravite** |
| --- | --- |
| 0 | Indéfini |
| 1 | Anomalie Bloquante |
| 2 | Anomalie grave |
| 3 | Avertissement |
| 4 | Anomalie rédhibitoire |

L'attribut simple gravite est de type Integer.

L'attribut simple labelGravite (attribut *dépendant* de gravite) est de type String.

#### Attribut simple suiteDonnee

Tout contrôle sait la suite à donner à l'objet contrôlé (fichier refusé, ligne retirée, champ mis à null, champ conservé, …). Le concept Suite A Donner ne sait rien du Contrôle qui l'a généré.

La suite à donner est donc un attribut simple (pas un objet)

Il est toujours plus facile de trier dans un tableau ce type de champs lorsqu'ils sont entiers numériques. En revanche, un label (String) est toujours plus lisible pour l'acteur. En outre, la MOA pourra le cas échéant modifier à postériori modifier le label à sa convenance.

Par exemple :

|  |  |
| --- | --- |
| **suiteDonnee** | **labelSuiteDonnee** |
| 1 | FICHIER ACCEPTE |
| 2 | FICHIER REFUSE |
| 3 | LIGNE CONSERVEE |
| 4 | LIGNE RETIREE |
| 5 | CHAMP ACCEPTE |
| 6 | CHAMP ANNULE |
| 7 | CHAMP MODIFIE |

L'attribut simple suiteDonnee est un Integer.

L'attribut labelSuiteDonnee (attribut *dépendant* de suiteDonnee) est une String.

#### Attribut Objet fichierResultant (File)

Tout Contrôle connaît le Fichier Résultant qu'il génère. Celui-ci peut être le même que le fichier entrant si le contrôle ne produit aucune modification, ou différent si le contrôle retire des lignes, annule des champs, …

Le Fichier Résultant ne sait rien du Contrôle qui l'a généré.

Il n'est donc pas nécessaire de créer un Type particulier (pure fabrication Objet) pour le concept de Fichier Résultant.

Le type File proposé par Java est suffisant.

#### Attribut Objet reglesGestion (Map<String, String>)

Tout Contrôle connaît les Règles de Gestion (RG) qu'il implémente. Il peut implémenter plusieurs Règles de Gestion.

Les Règles de Gestion ne connaissent pas les Contrôles qui les implémentent.

Il n'y a donc pas lieu de créer une pure fabrication (Objet spécifique) pour le concept Règle de Gestion.

Un Contrôle doit posséder sa **Collection** de Règles de Gestion qu'il implémente.

Une RG possède toujours un identifiant et une description. Par exemple :

| **Identifiant** | **Description** |
| --- | --- |
| RG-01-06\_ELIMINATION\_LIGNES\_VIDES | Aucun fichier ne doit comporter de lignes vides.  Si des lignes vides sont trouvées, elles sont éliminées et le programme continue. |
| RG-01-07\_ELIMINATION\_LIGNE\_REDHIBITOIRE | Aucun fichier ne doit comporter de lignes avec des champs "rédhibitoires" mal formés.  Si une telle ligne est trouvée, elle est éliminée et le programme continue. |

reglesGestion peut donc être une Collection (objet Java) de type Map<String, String> dans laquelle les enregistrements seront des couples <identifiant, description>

#### Attribut Objet reglesArt (Map<String, String>

Les Règles de l'Art peuvent être vues comme un cas particulier de règles de gestion.

Il n'est donc pas nécessaire de créer un attribut supplémentaire.

#### Attribut Objet rapportControleTxt

#### Attribut Objet rapportControleCsv

### Dictionnaire des données

| **Attribut** | **Type Java** | **description** |
| --- | --- | --- |
| **dateControle** | Date | Date d'exécution d'un contrôle |
| **userName** | String | Nom de l'organisation du User qui a déclenché le contrôle en soumettant un fichier (DIRA, DIRE, DARWIN, …) |
| **critere** | String | Critère vérifié par le contrôle comme par exemple "le fichier soumis n'est pas vide" ou "le champ Département dans la ligne du fichier HIT est renseigné". |
| **typeErreur** | Integer | Numéro indiquant le type de l'erreur détectée (1 pour champ non renseigné, 2 pour mauvais format de champ, …) |
| *labelTypeErreur* | String | Label associé au typeErreur (champ non renseigné pour 1, mauvais format de champ pour 2, …) |
| **actif** | boolean | Précise si le contrôle doit être exécuté (true) |
| **statut** | boolean | Précise si le contrôle a matché favorablement (true) ou défavorablement (false) |
| *labelStatut* | String | OK si le statut est true  KO si le statut est false |
| **gravite** | Integer | Gravité de l'anomalie recherchée par le contrôle (anomalie bloquante, anomalie rédhibitoire pour une ligne ou pour un champ, …) |
| *labelGravite* | String | Label de la gravité (Anomalie bloquante pour 1, anomalie grave pour 2, …) |
| **suiteDonnee** | Integer | Suite à donner à l'objet contrôlé si le contrôle est défavorable (2 pour FICHIER REFUSE, 4 pour LIGNE RETIREE, 6 pour CHAMP ANNULE, …) |
| *labelSuiteDonnee* | String | Label de la suite à donner à l'objet contrôlé si le contrôle est défavorable (FICHIER REFUSE pour 2, LIGNE RETIREE pour 4, CHAMP ANNULE pour 6, …) |
| **fichierResultant** | File | Fichier en sortie du contrôle comportant éventuellement moins de lignes, des champs annulés, … |
| **reglesGestion** | Map<String, String> | Collection contenant les Règles de Gestion et Règles de l'Art implémentées par le contrôle. |
|  |  |  |

### Etablir les relations inter-concepts (autres que Contrôle)

### Etudier l'enchaînement des contrôles

#### Nécessité d'enchaîner les contrôles

#### Importance de l'ordre des contrôles

### Eliminer les attributs non indispensables

### Modèle conceptuel des contrôles

## Tester le modèle conceptuel des Contrôles

# Conception des contrôles

## Extraire les attributs et méthodes livrées par le modèle conceptuel

## Factoriser au maximum le code (abstractions)

## Appliquer les règles de codage (Design Patterns)

## Définir les objets techniques

### Notion d'ordre du contrôle

Puisqu'il y a plusieurs enchaînements logiques des contrôles possibles, il peut être intéressant de décider de l'ordre dans lequel on choisira d'exécuter chaque contrôle.

Il est intéressant pour les développeurs (TMA) de pouvoir consulter dans quel ordre les contrôles sont exécutés lors de la vie réelle de l'application.

## Tester la réutilisabilité des contrôles

## "jouer" des contrôles

## Vérifier la Traçabilité Besoins-Exigences-RG-Classes-Attributs-Méthodes

## Bâtir le modèle de classe des contrôles

# Administration des contrôles dans l'application

# Capitalisation de la méthode