

Matthieu Jan  
Felix Bezançon

# Projet Dress'Class

## Intro

Dans le cadre de la matière "Analyse et conception des processus orientés services", nous avons eu à analyser et étendre un Système d'Information. Le projet, nommé Dress'Class, décrit une entreprise distribuée entre une Centrale d'achat et des Boutiques. Le SI de cette entreprise fait le lien entre ces acteurs.

Le client souhaite intégrer un nouveau scénario métier : le projet Rapid dress, permettant aux clients d'être servi plus rapidement, et demandant des fonctionnalités supplémentaires et une consolidation des processus existants.

L'objectif est d'identifier clairement le SI existant, identifier les problèmes actuels, les corriger puis intégrer les fonctions métier de Rapid dress.

## Table of Contents

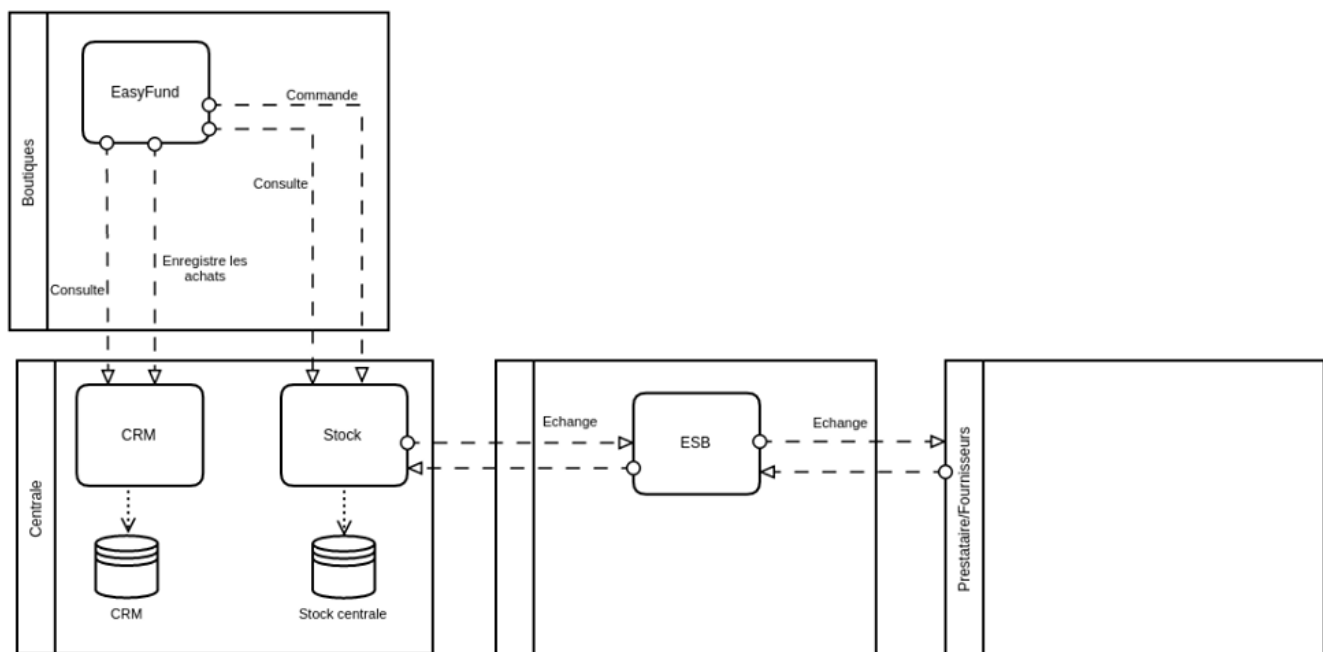
Intro.....	1
Cartographie du système existant.....	3
Vue d'ensemble.....	3
Centrale.....	4
Centrale – Prestataires/Fournisseurs.....	4
Boutiques – Centrale.....	5
Identification des problèmes existants.....	6
Problème 1 : Perte occasionnelle des commandes des boutiques vers la centrale d'achats.....	6
Problème 2 : Commande de produit qui n'existent plus.....	6
Problème 3 : Certains modèles ne répondant pas au critère de Rapid commande sont commandés même s'ils n'existent pas dans le stock.....	6
Problème 4 : Commande de Central au Fournisseurs parfois perdues.....	6
Problème 5 : Achats clients parfois perdues et ignorés par le programme fidélité.....	7
Problème 6 : Certains modèles présent sur le site n'existent plus dans le stock .....	7
Synthèse des problèmes.....	7
Modélisation des processus métier.....	8
Préambule.....	8
Modèle du processus métier.....	9
Tableau de correspondance activité application.....	9
Mise en oeuvre de Rapide Dress et du site web.....	9
Identification des objets métier et des services.....	10
Identification des objets métier.....	10
Identification des services.....	11


# Cartographie du système existant

Faire une description, la plus exhaustive possible, avec les informations dont l'on dispose (Sujet du TP 1 - page 3) afin de modéliser fidèlement le SI. On réalise cette modélisation avec Camunda-Modeler, avec le format BPMN.


## Vue d'ensemble

Voici la représentation, après analyse, de notre modèle. Il est à noter que le comportement interne de Prestataires/Fournisseurs n'est pas visible, un ESB gérant les échanges (représenté ici à l'extérieur du SI).



 : Base de données

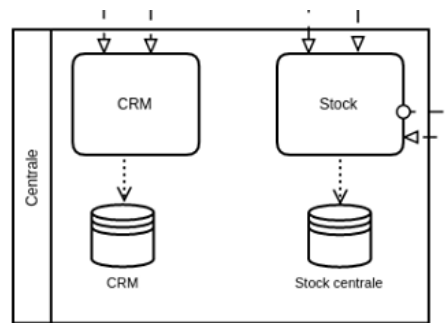
 : Application

 : Acteur du SI

# Centrale

L'acteur Centrale est la représentation du SI qui traite les processus dédiés à la centrale d'achats. Il comprend deux applications :

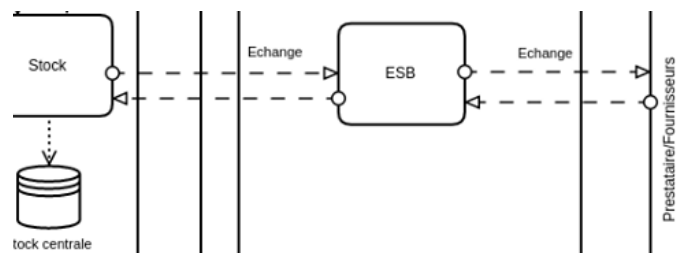
- CRM : centralise les informations client, s'appuie sur une base de données.
- Stock : centralise l'inventaire, gère le stock, reçoit et traite les commandes. Il s'appuie également sur une base de données.



## Centrale - Prestataires/Fournisseurs

Centrale communique avec les prestataires et fournisseurs par l'intermédiaire d'un Enterprise Service Bus. Le fonctionnement et les applications de ces acteurs ne sont pas renseignés (et n'ont pas besoin de l'être).

- Stock::echange::ESB et ESB::echange::Stock représente les informations échangées par Centrale via Stock vers les partenaires. Cette opération génère un flux public, BtoB. On suppose que ce flux est cadencé et de masse, mais le document ne le précise pas.

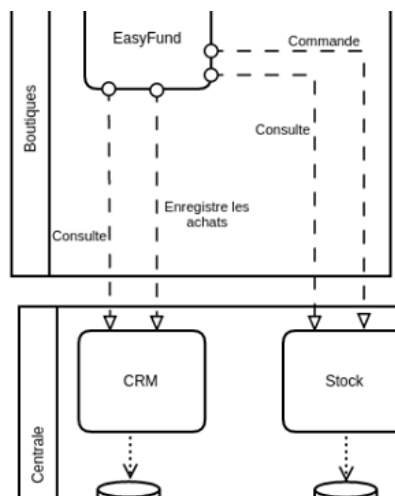


## Boutiques - Centrale

Les acteurs Boutiques représente le SI des boutiques franchisées, qui traite des commandes et des informations client. Pour Centrale, voir dessus.

Boutiques comprend une application, EasyFund, qui a plusieurs opérations :

- EasyFund::consulte::CRM, consulte la fiche fidélité de ses clients dans le CRM lorsque ceux ci passent en caisse. Cette opération génère un flux public, AtoA, cours de l'eau et unitaire. On suppose que ce flux est synchronisé, la demande de fiche étant utilisé dans l'immédiat pour faire une remise au client.
- EasyFund::enregistre::CRM, enregistre les achats effectué par les clients auprès du CRM, qui le stocke dans la base de données. Cette opération génère un flux public, AtoA, cours de l'eau et unitaire.
- EasyFund::consulte::Stock, consulte le stock d'un produit qu'un client souhaite acheter afin de vérifier sa disponibilité, ainsi que les modèles et tailles disponibles. Cette opération génère un flux public, AtoA, cours de l'eau et unitaire.
- EasyFund::commande::Stock, renseigne l'application Stock sur une commande effectué, lui déléguant la suite du processus d'achat. Cette opération génère un flux public, AtoA, cours de l'eau et unitaire.



# Identification des problèmes existants

L'objectif de cette partie est d'identifier les problèmes du SI en fonction des symptômes visibles. Les documents ne fournissant pas assez d'information pour établir un réel diagnostic, nous émettrons à chaque une hypothèse valable, que nous considererons comme LE problème rencontré.

## **Problème 1 : Perte occasionnelle des commandes des boutiques vers la centrale d'achats.**

Le flux EasyFund::commande::Stock est en cause. Nous pouvons emmettre une hypothèse.

- Si ce flux est asynchrone, il est possible que Stock ne fournissent pas d'acquittement lors de l'émission d'une commande. Ainsi, en cas d'erreurs du réseau ou de l'application conduisant à la perte du message, celui ci n'est pas re-émis.

## **Problème 2 : Commande de produit qui n'existent plus**

Les flux EasyFund::consulte::Stock et EasyFund::commande::Stock sont en cause.

- Si EasyFund conserve les informations qu'il récupère depuis Stock et ne demande que des mises à jour, il est possible que les flux soient mal cadencés et que des erreurs apparaissent.

## **Problème 3 : Certains modèles ne répondant pas au critère de Rapid commande sont commandés même s'ils n'existent pas dans le stock**

Nous n'avons pas compris la phrase, on suppose que le problème est similaire au problème 2.

## **Problème 4 : Commande de Central au Fournisseurs parfois perdues**

Le flux Stock::echange::ESB est en cause. Nous emettons une hypothèse similaire au problème 1.

- Si ce flux est asynchrone, il est possible que ESB ne fournissent pas d'acquittement lors de l'émission d'une commande. Ainsi, en cas d'erreurs du

réseau ou de l'application conduisant à la perte du message, celui ci n'est pas re-émis.

## **Problème 5 : Achats clients parfois perdues et ignorés par le programme fidélité**

Le flux EasyFund::enregistre::CRM est en cause. Nous émettons une hypothèse similaire au problème 1 et 4.

- Si ce flux est asynchrone, il est possible que CRM ne fournissent pas d'acquittement lors de l'enregistrement d'un achat. Ainsi, en cas d'erreurs du réseau ou de l'application conduisant à la perte du message, celui ci n'est pas re-émis.

## **Problème 6 : Certains modèles présent sur le site n'existent plus dans le stock**

Le (nouveau) flux Web::consulte::Stock est en cause. Nous pouvons émettre une hypothèse.

- Si Web conserve les informations qu'il récupère depuis Stock et ne demande que des mises à jour, il est possible que les flux soient mal cadencés et que des erreurs apparaissent.

## **Synthèse des problèmes**

Après analyse, on constate deux types de problèmes récurrent :

- Les flux ne sont pas synchronisé ou, à défaut d'être synchronisé, n'acquittent pas les réceptions de messages.
- Les vues qu'ont les applications du stock n'est pas ou mal synchronisé, conduisant à des erreurs.

# Modélisation des processus métier

## Préambule

Plusieurs points sont à noter pour une bonne compréhension du processus, points que nous avons dû imposer pour proposer un processus complet et exhaustif :

- Le client doit être inscrit pour faire une commande, vu que son profil sera demandé lors du retrait du produit.
- L'inventaire est géré et rempli par la centrale d'achat indépendamment des commandes effectuées. ( Les fournisseurs ne sont pas intégrés au processus d'achat).
- Le retrait des produits se fait en boutique (même depuis le site web).
- Le paiement de la commande se fait lors du retrait (étant donné les problèmes précédents, qui ont dû faire un peu de bruit, on ne peut pas vraiment se permettre de faire payer le client en avance).
- Le signal capturé dans le diagramme par le client n'est émis nulle part, car il peut se produire à tout moment, et nous n'avons pas voulu surcharger le diagramme en le plaçant à chaque étape.
- De la même façon, lors de l'attente de message, un timeout s'enclenche systématique, et génère une erreur si il arrive au bout.
- Les problèmes identifiés précédemment ont été corrigés en forçant l'envoi de message lors d'un envoi de message, et une consultation systématique de Centrale lors de l'accès à la base de données.



## Modèle du processus métier

Voir Processus\_Existant.png ou Processus\_Existant.bpmn

## Tableau de correspondance activité application

Voici ici le tableau de correspondance des processus de notre modèle avec les applications qui les exécutent.

**Processus Client** : L'ensemble des processus client sont des réflexions ou discussions avec le vendeur en direct.

**Processus Boutique**: L'ensemble des processus boutique sont soit lié à EasyFund, soit des discussions avec le client.

### **Processus Centrale :**

- Vérifier les identifiants → CRM
- Créer un nouveau client → CRM
- Vérification du stock → Stock
- Mise à jour du stock → Stock
- Valider et enregistrer la commande en attente de point retrait → Stock puis CRM
- Valider la commande → Stock
- Annulation de la commande → Stock et CRM
- Mise à jour du stock (après annulation) → Stock

## Mise en oeuvre de Rapide Dress et du site web

La mise en oeuvre de la fiabilité de Rapid Dress a été faite dans le premier processus, lors de la correction des erreurs

Voir Processus\_Cible.png ou Processus\_Cible.bpmn

# Identification des objets métier et des services

Dans la poursuite de ce projet, nous allons maintenant devoir identifier les objets métier et les services liés aux processus d'achat que nous avons modélisé précédemment.

## Identification des objets métier

Nous allons commencer par identifier les différentes données traitées par le SI. On en distingue trois indépendante :

- ReferenceClient, caractérisé par un identifiant.
- ReferenceProduit, caractérisé par un nom, un prix et une quantité disponible.
- ReferenceGéographique, caractérisé par une adresse et un statut. Il représente un lieu utilisé dans le SI. Dans notre cas, nous ne traitons que des points de retraits, qui auront dans le statut correspondant.

De plus, le SI traite certaines données composites ou bien décoré par d'autres attributs :

- Produit, caractérisé par une "ReferenceProduit" et une quantité. Utilisé par exemple dans les commandes, la gestion de l'inventaire etc ...
- Commande, caractérisé par une liste de "Produit", une "ReferenceClient" et une "ReferenceGéographique avec le statut point de retrait".
- FidClient, caractérisé par une "ReferenceClient" et la liste de "Commande avec cette ReferenceClient" et les points qui lui sont associés.

## Identification des services

Pour chacune de ces données on distingue des services qui les fournissent ou les consomment. Nous allons ici identifier et caractériser uniquement les services nécessaires au processus d'achat.

### **Utilisateur.service :**

Utilisateur.service sert à fournir des informations concernant les utilisateurs du SI. Il permet d'utiliser les actions CRUD pour manipuler la base de données utilisateur.

Relation :

Requiert : Utilisateur.bdd

Fourni : Connexion.service / Inscription.service / Commande.service / Fidelite.service

### **Catalogue.service :**

Catalogue.service sert à fournir des informations concernant les références produits connus du SI. Il fourni des objets ReferenceProduit

Relation :

Requiert : Catalogue.bdd

Fourni : Commande.service / Inventaire.service

### **Geographie.service :**

Geographie.service sert à fournir des informations concernant les lieux et emplacements connus du SI. Il fourni des objets ReferenceGeographique

Relation :

Requiert : Geographie.bdd

Fourni : Commande.service / SiteWeb.app EasyFund.app

Ces trois services fournissent des informations relativement statiques, peu souvent modifié mais indispensable au fonctionnement du SI. Ces services ne sont pas publiés aux applications mais uniquement à des services qui les utilisent.

**Connexion.service :**

Connexion.service sert à authentifier les utilisateurs du SI.

Relation :

Requiert : Utilisateur.service

Fourni : SiteWeb.app / EasyFund.app

**Inscription.service :**

Inscription.service sert à créer de nouveaux utilisateurs dans le SI.

Relation :

Requiert : Utilisateur.service

Fourni : SiteWeb.app / EasyFund.app

**Commande.service :**

Commande.service sert à fournir et enregistrer des informations concernant les commandes faites.

Relation :

Requiert : Commande.bdd / Utilisateur.service / Catalogue.service / Inventaire.service / Geographie.service / Fidelite.service

Fourni : SiteWeb.app / EasyFund.app / Fidelite.service

**Inventaire.service :**

Inventaire.service sert à fournir et enregistrer les informations concernant le stock de Centrale.

Requiert : Inventaire.bdd / Catalogue.service

Fourni : Commande.service / EasyFund.app / SiteWeb.app

**Fidelite.service :**

Fidelite.service sert à fournir et enregistrer les informations concernant le compte fidélité d'un client.

Requiert : Fidelite.bdd / Utilisateur.service / Commande.service

Fourni : EasyFund.app / SiteWeb.apCp / Commande.service