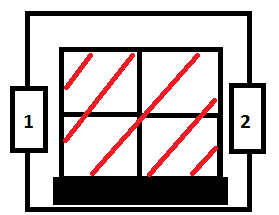
**Information système Fleury Michon**

**Présentation de l’entreprise et de son fonctionnement.**  
  
 **MES** 🡪 Manufacturing Execution System (=Système d’exécution de production).  
Exécute des ordres de réception, fabrication (consommation et production), traçabilité, expédition, suivi de performance et gestion des stocks.  
  
Quelques abréviation :   
TLM – Traiteur de La Mer (site de Chantonnay)  
TLS – Traiteur Libre-Service  
MTT – Mont Tiffauges Traiteur  
MTJ - Mont Tiffauges Jambon  
CH – Charcuterie  
OF – Ordre de Fabrication  
UT – Unité de Traçabilité

à rajouter : calcul de besoin (Gestion production)  
 sens de l’information  
 remonte l’information  
 plan de production, réception et d’expédition (M E S)  
  
  
  
  
  
  
  
  
  
  
Contexte du projet : TLM.  
  
  
à rajouter : flèche étiquetage  
  
Note : *À Cambrai, l’étiquetage se fait au suremballage.  
 À Mouilleron, il ne s’agit pas d’un étiquetage mais d’un marquage au niveau du conditionnement.*  
  
Étiquetage possible au suremballage mais surtout étiquetage sur chaque colis durant la palettisation.



à rajouter : 1 & 2 étiqueteuses  
 cartons à étiqueter  
 palette sur tapis  
 automate  
  
PRODUCIM --> MesToColis --> affichage des ordres de fabrication, des données à imprimer ainsi que du transfert de données.  
  
Sujet du stage : supprimer iDaro et développer un logiciel le remplaçant.  
  
[Schéma du principe de fonctionnement avec iDaro]  
  
De plus, quatre améliorations sont à voir :   
1 - L’étiqueteuse VideoJet surnommé ‘Mulet’ à gérer (protocole de communication Zipher text).  
2 - Fichier de paramétrage à remettre en xml.  
3 - Envoie des données de différentes manières (1 VJ, 1 automate, 1 automate + 1 VJ etc).   
4 - Ligne de suremballage (L-98) avec 1 automate et 1 VJ.  
5 - Portique de palettisation (L-99).  
  
**Présentation du système informatique et de l’environnement de développement.**  
 **Référentiel** 🡪 Dossier contenant les différents projets fonctionnels.

Dans le cadre de mon projet, on y retrouve Les dossiers suivants :  
Référentiel/FleuryMichon/MesToColis/ Distrib  
 Package  
 Librairie  
 Source  
  
Le dossier Distrib contient les fichiers d’installations de la version actuelle de MesToColis.  
Le dossier Package regroupe les différentes versions fonctionnels de l’application. Lors du lancement de l’une d’elles, le fichier Distrib est automatiquement mis à jour pour que l’installation se fixe sur la version choisit.  
Le dossier Librairie contient l’ensemble des librairies, Framework, dll… etc.  
Le dossier Sources contient plusieurs sous-dossier :  
 - VB.NET : l’ensemble des versions de Visual Studio ainsi que le fichier solution de FleuryMichon.MesToColis.IHM.sln.  
 - SQL : Tous les scripts utiles à la création de la base de données.  
 - Documentations : Contient deux fichiers important dans l’avancée du projet (history.txt et Doc). Doc étant un raccourci vers le dossier des documentations du projet MesToColis et history.txt un fichier texte contenant l’intégralité des modifications datés.  
  
 **Oracle SQL DEVELOPER** 🡪 Il s’agit du logiciel permettant de gérer les bases de données de l’entreprise. On a alors la possibilité pour une table donnée d’observer les tables, vues, procédures… etc. Il existe les BDD PMPx et PMTx (le P pour Production, T pour Test et le x (allant de 1 à 9) qui est l’identifiant de la BDD pour un site donnée). Dans le cadre de notre projet, nous travaillerons avec la BDD PMT1 puisque c’est celle qui est utilisé pour réaliser des tests sur le site de TLM.  
On doit alors au tout début créer une nouvelle connexion, pour cela, il suffit de faire :  
Clic droit – Nouvelle connexion – Saisir un nom de connexion, un utilisateur (fm) et un mot de passe (fm) – Sélectionner le type de connexion TLS ainsi que la BDD sur laquelle on souhaite se connecter – Enregistrer et se connecter.  
  
 **Visual Studio** 🡪 Il s’agit d’un logiciel de développement pour coder en VB.NET. Outre les différentes fonctionnalités du logiciel, on a observé de plus près les fichiers qui composaient l’application MesToClient.  
On retrouve donc : My Project -> paramètres de lancement de l’application  
 [codemachine;mode;droits]  
 Business / Data / Entities -> ~ modèle DAO en JEE  
 Configuration, Images, ModelView -> …  
 UserControl -> objets de l’IHM  
 AppConfig -> configuration connexion BDD / machine-étiqueteuse  
 MainWindow.xaml -> code de l’IHM  
 MainWindow.xaml.vb -> IHM physique  
  
Les masques d’étiquette : Elles sont différentes en fonctions des produits. (ex : poulet, poisson… etc).  
Les #ref# qui composent le fichier XML des masques correspond à un identifiant dans la BDD (ex : GP000043).   
Dans le code de l’application, c’est dans les fichiers ‘’Provider‘’ que les évènements (clique, check… etc) sont lancés et c’est donc à ce moment-là que pour une certaine action, l’affichage des paramètres d’un produit/étiquette est mises à jour.  
  
**Résultat de la réunion iDaro.**

Objectif :

Redévelopper en interne Fleury Michon la communication entre le MES et les étiqueteuses VIDEOJET lors de la palettisation, sur la L98 et sur le mulet.

Solution technique :

* Communication de toutes les infos du MES à l'automate
* L'automate enverra les données à étiqueter aux Videojet
* Vérification par l'automate que les données lues par le lecteur CAB sont bien celles transmises par le MES
* Gestion d'un flag pour tests de communication entre le MES et l'automate :
  + Lors de l'envoi des données le MES met à jour un flag à 0
  + Lors de la récupération des données l'automate repasse le flag à 1
* Pour la L98 : au déclenchement d'OF : envoyer le code de l'OF (code unique) afin que l'automate contrôle que les données étiquetées sont bien celles de l'OF
* Automate = Schneider M340

Tests :

* Sébastien : mettre à disposition de Quentin, dans le bureau de Kévin, un automate de test avec un afficheur. Ils seront branchés sur le réseau.

Priorité :

=> Déploiement sur le Mulet avant l'été

A faire :

* DSI :

À faire Envoyer à Sébastien la liste de toutes les infos qu'on envoie à l'automate et aux Videojet

À faire Envoyer exemple de trame + doc Zipher

* Sébastien :

À faire Envoyer les zones mémoires

À faire Mise à dispo banc de test

**Résultat mise au point Pierrick Réau.**  
 Suite à la réunion au sujet du service iDaro, la priorité est de faire fonctionner l’étiqueteuse « Mulet ». Pour cela, la première tâche du projet sera de modifier l’application MesToColis afin qu’elle envoie directement les données sur l’automate qui est chargé de communiquer avec étiqueteuse (date limite fixée à fin mai). Cette modification sera ensuite adaptée sur l’ensemble de la ligne L-99 (2 étiqueteuses + « Mulet » donc utilisation de deux espaces mémoires différents de l’automate) ainsi que sur la ligne L-98 (2 étiqueteuses). La communication entre l’automate et les étiqueteuses sera ensuite géré par M. Marais Sébastien via l’utilisation du protocole Zipher text.  
Pour réaliser cette étape du projet, on m’a présenté deux applications qui communiquent avec un automate (une en lecture et une en écriture) pour voir le fonctionnement ainsi que les différentes fonctions utilisées par celles-ci. Enfin, on m’a aussi montré les logiciels PC Network Interface (qui permet de voir l’ensemble des automates qui sont connectés au réseau) et OPC Client (qui permet de lire les informations envoyé à un emplacement mémoire d’un automate). Ces deux logiciels sont disponible sur le serveur 94 - SRVMESACQDEV dans le répertoire : DisqueC/ProgrammeFiles(x86)/BradCommunications/PCNetworkInterface/4.1  
  
Le principe de fonctionnement d’un automate est la suivant, il possède un nom (identifiant), une adresse IP ainsi que des « mots » (= cases mémoires). Dans l’un des fichiers de configuration de l’application, on retrouve celles du/des automate(s) visé(s) ainsi que l’ensemble de ces mots reliés alors à des données du programme. Il faut alors communiquer son nom qui pourra ensuite être utilisé comme « Objet » à l’intérieur du code. Au préalable, pour gérer l’intégralité des automates visés, il faut utiliser le logiciel PC Network Interface afin de les référencier dans le réseau. Il sera alors possible d’utiliser le logiciel OPC Client pour lire les informations envoyé à l’issue de nos tests.  
Pour cela, il suffit de lancer le logiciel, Server -> Connect to server… -> Choisir APPLICOM.OPCServer.1  
Puis il faut ajouter un nouveau groupe et valider. Enfin, il faut ajouter un item (automates) et bien veiller à choisir l’emplacement du bon « mot » à lire.  
Pour ce qui est des applications à analyser et utiliser, elles se situent dans les répertoires Sources de MesToTrk (écriture vers l’automate) et ACQMoulageToMes (lecture de l’automate).

Note : les tests à distance pourront être fait sur l’automate DB1 qui un émulateur d’automate (toujours accessible).  
  
**Développement application applicomDB1.**  
  
Différence entre la fonction Read() et SyncRead() :   
Pour la fonction SyncRead(), le client OPC ordonne au serveur OPC d'aller lire les données du périphérique (c'est-à-dire l'automate). Le client OPC attend que le serveur OPC retourne les données. À moins d'exécuter de manière multi-thread (ce que font de nombreux clients, y compris le contrôle de données OPC ActiveX), le client OPC ne peut rien faire d’autre en attendant et cela même si le client OPC est multi-thread (donc gaspillage de ressources informatiques)  
Pour la fonction Read(), le client OPC (c'est-à-dire le contrôle OPC ActiveX Data Control) indique au serveur OPC les noms des éléments OPC (item) auxquels il souhaite des mises à jour et la fréquence à laquelle il veut des mises à jour, puis le client retourne à ce qu'il fait. Chaque fois que le serveur OPC a de nouvelles données à signaler, il fait un "rappel" pour le client OPC et délivre les données.   
Différence entre la fonction Write() et SyncWrite() :