



Grenoble INP – ENSIMAG
Ecole Nationale Supérieure d'Informatique et de Mathématiques Appliquées

Rapport de Projet de Fin d'Etudes

Effectué chez Amundi en tant que prestataire Aubay

Titre du Sujet de Stage

André Mathilde
3A – Graphics, Vision and Robotics
02 Mars 2015 – 31 Août 2015

Amundi
90 Boulevard Pasteur
BP XX
75015 Paris

Responsable de stage
Nom Et Prenom Tuteur Entreprise
Tuteur de l'école
Rippert Christophe

Table des matières

1	Résumé	3
2	Introduction	4
3	Présentation des entreprises	5
3.1	Aubay	5
3.2	Amundi	5
3.3	Mon stage au sein de l'entreprise	5
4	Problématique et objectifs	7
4.1	Contexte du projet	7
4.1.1	Le cadre : AMEX	7
4.1.2	Utilisation des FlowDetails	8
4.2	Travail à réaliser	10
4.3	Objectifs précis	10
5	Les solutions techniques	12
5.1	Syntheses de l'existant	12
5.1.1	Langage de développement et concepts de programmation	12
5.1.2	Les outils utilisés	13
5.2	Description du travail	13
5.2.1	Analyse de l'existant	13
5.2.2	La nouvelle modélisation	14
5.2.3	Migration de l'existant	16
5.2.4	Refonte des IHM	17
5.2.5	Bascule sur les nouveaux services	18
5.3	Protocole d'évaluation	18
6	Avancement du projet	19
7	Impressions personnelles	20
8	annexe	21
8.1	Lexique	21

Chapitre 1

Résumé

En dernière année de cycle ingénieur à l'ensimag Grenoble, les étudiants sont amenés à effectuer un stage en entreprise d'une durée de six mois.

J'ai effectué mon stage de Mars à Aout 2015 pour la Société de Service en Ingénierie Informatique (SSII) AUBAY, en mission chez leur client AMUNDI.

J'ai été amenée à travailler autour d'un logiciel interne à AMUNDI, nommé AMEX pour Amundi Exchange Message. J'ai fait parti de l'équipe MFN (Middle Flux Négociation) qui regroupe une vingtaine de personnes et dont le responsable est Jean-François Morin. Cette équipe gère plusieurs applications dont la plateforme internationale Amundi Exchange Message, Amex qui permet de transmettre des messages intra applicatifs et inter banques suivant des formats standards entre les différents partenaires. Ce projet est principalement pris en charge par Pierre Chatelier, qui m'a suivi et encadré tout au long de ma mission.

Chapitre 2

Introduction

Mon stage s'est déroulé chez Amundi, dans l'équipe Middle Flux (MFX). Ce document décrit en détails mon projet de six mois au sein de cette équipe.

Le but de ma mission est de réaliser une refonte d'un des composants du logiciel Amex. Cela s'est déroulé en plusieurs étapes, la découverte des technologies AMUNDI et de la plateforme AMEX, puis la réalisation du projet a pu commencer. Tout d'abord, il a fallu effectué une analyse de la modélisation existante afin d'en proposer une nouvelle sous forme d'un diagramme de classe. Une fois cette nouvelle modélisation validée par différentes personnes, l'implémentation base de données et JAVA a eu lieu. J'ai donc pu me charger du développement d'un module du logiciel et d'impacter l'existant (IHM, flux et API).

Au cours de ce rapport, je vais tout d'abord vous présenter AUBAY l'entreprise m'ayant fourni ce stage ainsi que son client Amundi chez qui je travaille. Dans un second temps, les problématiques et objectifs du projet au sein de l'entreprise seront développés, vous pourrez ainsi lire une description du logiciel sur lequel je travaille. Je vous décrirai ensuite les solutions techniques et l'avancement de mon projet tout au long du stage, pour finir avec mes impressions personnelles.

Chapitre 3

Présentation des entreprises

3.1 Aubay

3.2 Amundi

Amundi est une société de conseil en épargne salariale, née du partenariat du groupe Crédit Agricole SA (actionnaire à 80%) et de la Société Générale (actionnaire à 20%). Cette entreprise de gestion des actifs compte parmi les plus grands acteurs de l'industrie de l'asset management mondial.

Un Asset Manager crée et gère au quotidien des produits de placements, à savoir les OPCVM (Organisme de Placement Commun en Valeurs Mobilières). Les particuliers et les entreprises souhaitant confier leur argent pour qu'il soit géré peuvent faire appel à un Asset Manager.

3.3 Mon stage au sein de l'entreprise

Dans le cadre de mon projet de fin d'étude, j'ai intégré le service informatique d'Amundi et plus précisément l'équipe MFX, pour Middle Flux, dirigée par Jean-François Morin.

Le domaine MFX a en charge le développement et la maintenance des applications utilisées par le Middle Office Flux Amundi. Celui-ci se décompose en deux équipes, la maîtrise d'ouvrage (MOA) et la maîtrise d'Oeuvre (MOE). Je fais partie de cette dernière.

Dans le domaine on va distinguer quatre types d'applications :

1. Applications permettant d'assurer le cycle de vie des transactions : ETC (Electronic Trade Confirmation) pour le Matching Broker et ETD (Electronic Trade Delivery) pour le règlement-livraison (le dépositaire et le valorisateur).
2. Application permettant d'assurer le cycle de vie des titres financiers (Intégration des opérations sur titres) : MOCA, Middle Office Corporate Action.

3. Application permettant l'intégration des collectes des souscription-rachats sur les fonds Amundi : GSR, Gestion des souscriptions-rachats.
4. Une plateforme d'échange AMEX dont le périmètre va au delà du domaine MFX, mais dont le développement et la maintenance du socle technique est de la responsabilité du domaine MFX.

Amundi gère certaines applications qui ont besoin de communiquer entre elles en envoyant et recevant de nombreux messages. Ces derniers transitent à travers une même application appelée Amex. J'ai travaillé sur la plateforme Amex et plus précisément sur une refonte totale d'un de ses composants, les FlowDetails. Le but principal de cette refonte pour les équipes utilisant AMEX est une amélioration des performances.

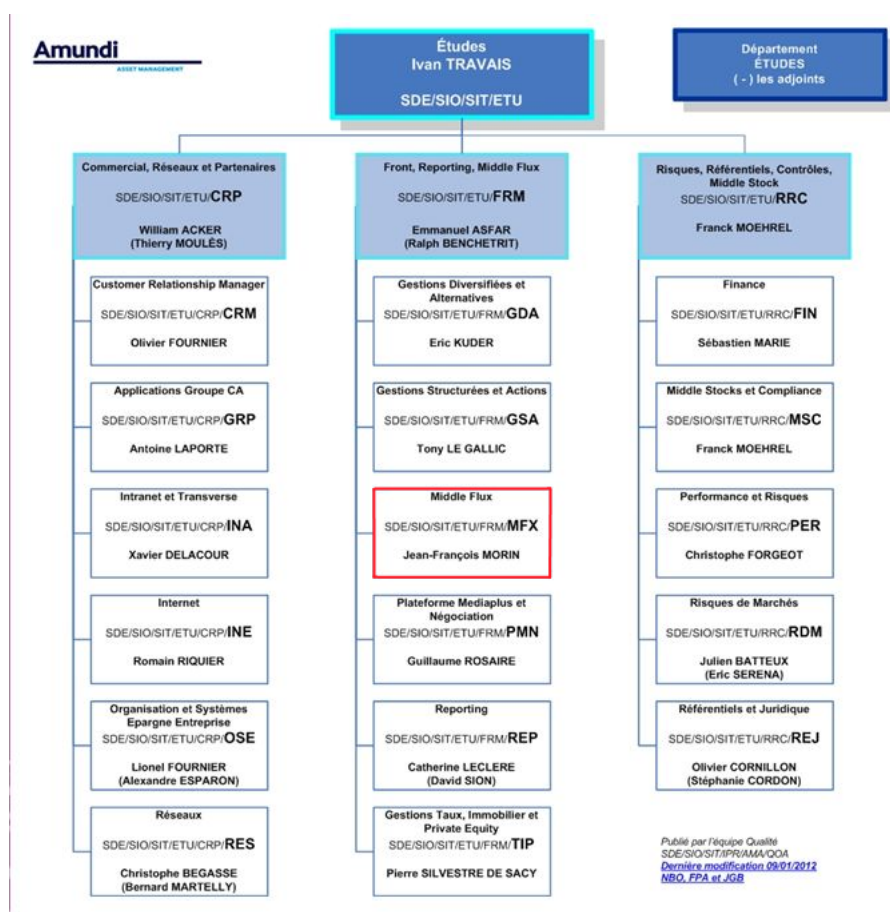


FIGURE 3.1 – Organigramme de Amundi IT Services

Chapitre 4

Problématique et objectifs

4.1 Contexte du projet

4.1.1 Le cadre : AMEX

L'application AMEX est une plateforme d'échange de messages pouvant gérer aussi bien des échanges inter-applicatifs au sein du système d'information que des échanges avec des applications externes de partenaires d'Amundi. C'est une plateforme multi-format et multi-canal. Elle permet l'échange de messages sous différents formats, ceux-ci pouvant être standardisés ; (le plus connu est SWIFT, format créé par un partenariat des plus grandes banques mondiales) ou propriétaires c'est à dire développé spécialement pour communiquer avec un partenaire spécifique. Les messages peuvent être envoyés via différents supports (JMS, Web Services, Fax, Fichier, File MQ, Imprimantes, Mail).

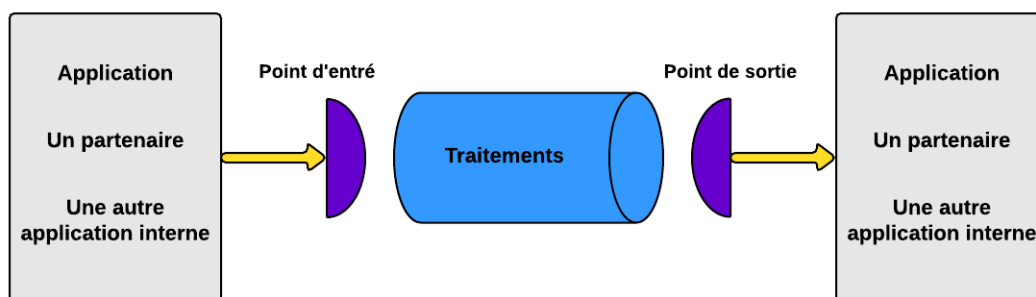


FIGURE 4.1 – Les flux Amex

Amex regroupe une centaine de flux. Ils sont utilisés notamment pour la conversion d'un message d'un format X vers un format Y. Cela peut se faire via un format d'échange pivot : le STPML qui est créé à partir du XML. Les différentes API utilisées par les flux Amex permettent, par exemple, d'envoyer des mails, des fax, d'auditer les messages en base de

données ..

Amex a un service d'Audit permettant de sauvegarder tous les messages transitants par cette plateforme.

Un flowdetail est un composant technique décrivant l'entrée ou la sortie d'un flux de communication. Ainsi lors de la création d'un nouveau flux, l'utilisateur crée deux Flowdetails, un représentant l'entrée du flux (input) et un autre représentant sa sortie (output). Un flowdetail contient des informations permettant de connaître l'émetteur et le destinataire du message ainsi que son format, média (d'entrée ou de sorti). Ce composant est nécessaire pour le routage du message mais aussi pour l'audit. L'image ci-dessous montre les informations regroupées dans un FlowDetails.












	flow_id	integer(10)
	flow_ref	char(255)
	media	char(255)
	format	char(255)
	sub_format	char(255) N
	rec_cre_user	char char(255)
	val_start_date	date
	val_end_date	date
	label	char(255)
	comment	char(255) N
	schedule_id	integer(10)

FIGURE 4.2 – Un FlowDetails

4.1.2 Utilisation des FlowDetails

Un logiciel développé en interne chez Amundi appelé MediaPlus Alto propose un module permettant de gérer les FlowDetails : les visualiser, les créer, les supprimer et les modifier. Mais d'autres modules utilisent aussi les FlowDetails à travers des arbres Rule-Solver notamment. Ces modules offrent la possibilité d'éditer des arbres de règles et de les sauvegarder. Un arbre RuleSolver peut par exemple, servir à retourner la liste des tâches à effectuer selon tel ou tel type.

Ces modules sont utilisés par des personnes de la MOE principalement pour des tests et par des personnes de la MOA lors de la création de nouveaux flux. Le but de mon stage est de faciliter le travail de ces personnes en proposant une implémentation plus efficace des FlowDetails. La première image ci-dessous est la vue du logiciel, permettant de visualiser ou de créer des FlowDetails.

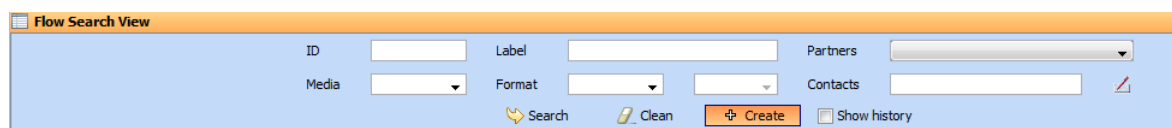


FIGURE 4.3 – Visualisation des FlowDetails

La seconde image est le résultat s'affichant après avoir cliqué sur le bouton 'Create' et choisi 'Mail' comme média. L'utilisateur doit choisir un Format, plusieurs contacts mail (FROM, TO). Cliquer sur le bouton 'Save' créera le FlowDetail.

The screenshot shows a web form titled 'Flow' with a light blue header. The form is divided into three main sections: 'Flow', 'Contacts', and 'Comment'.
1. 'Flow' section: Contains input fields for 'Ref.' and 'ID.', a 'Label' field, a 'Media' dropdown menu with 'MAIL' selected (highlighted in orange), a 'Format' dropdown menu, and a 'SubForma' dropdown menu. There are also two small square icons to the right of the 'Format' dropdown.
2. 'Contacts' section: Contains 'From' and 'To' email address fields, each with a blue envelope icon and a green plus icon. Below these are links for 'Add Cc' and 'Add Bc'. A 'Subject' field and a larger 'Body' text area follow.
3. 'Comment' section: A large text area for additional notes.
At the bottom right of the form are 'Save' and 'Cancel' buttons.

FIGURE 4.4 – Creation d'un flowdetail Mail

4.2 Travail à réaliser

Mon stage de six mois consiste en une refonte totale du composant FlowDetails. Le but principal étant d'améliorer les performances notamment lors de la création des FlowDetails mais aussi de permettre une modélisation plus claire et évolutive. Lors de la création d'un FlowDetail à partir de l'interface, les étapes sont très lentes, avant qu'une fenêtre s'affiche pour la première fois, il faut souvent attendre plusieurs secondes, ce qui rend le travail de certaines personnes nettement moins efficace. Le premier développement d'Amex a été réalisé il y a ... ans. Les besoins et la quantité de flux et de messages échangés ont beaucoup augmenté depuis. Par exemple, des informations sur des contacts sont enregistrés dans une base de données. Ces informations ont évolués au fil du temps, par exemple si un message de format SWIFT transite dans AMEX, deux informations sont nécessaires : le BIC du contact ainsi que son DN distinguish name. La modélisation de départ n'étant pas très générique un deuxième attribut a donc été ajouté pour tous types de contacts (Mail, Jms, Printer, SWIFT...), mais celui-ci a une valeur nul pour tous les autres que SWIFT. Ceci était un problème à résoudre, et ainsi de rendre la modélisation plus évolutives pour d'éventuels changements futurs.

Afin d'effectuer une refonte de ce composant, mon travail a été divisé en trois principales étapes.

La première étape a été d'analyser l'existant afin de comprendre la modélisation actuelle des FlowDetails et les différents services utilisant ce composant. Le but est ensuite de proposer une nouvelle modélisation base de données de ces composants. Dans un deuxième temps, il faudra implémenter et mettre en place la nouvelle modélisation et ainsi adapter les interfaces Homme Machine (IHM). Le client d'administration des FlowDetails devra répondre aux besoins des MOA. La dernière étape consiste en l'exposition de ces services d'administration sous forme de service REST afin de permettre un découplage applicatif nécessaire aux évolutions du middleware.

Le diagramme de Gantt ci dessous présentes les différentes étapes. Voir en annexe pour connaître les dates d'échéance qui ont été fixées en début de stage.

Mon travail a été réalisé suivant un cycle précis permettant d'aboutir à une validation de chacune des étapes.

4.3 Objectifs précis

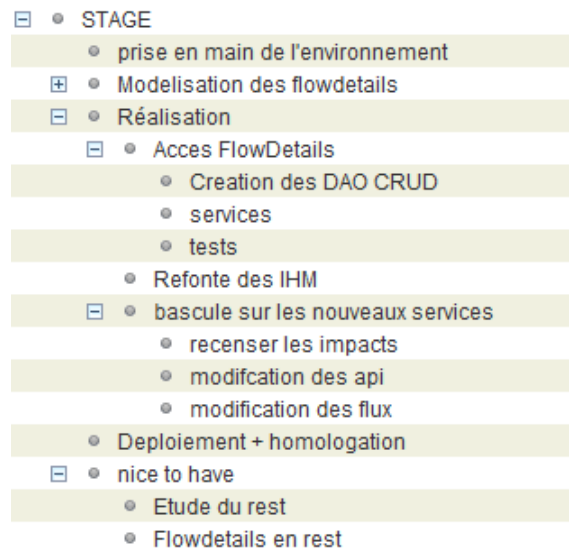


FIGURE 4.5 – Diagramme de Gantt

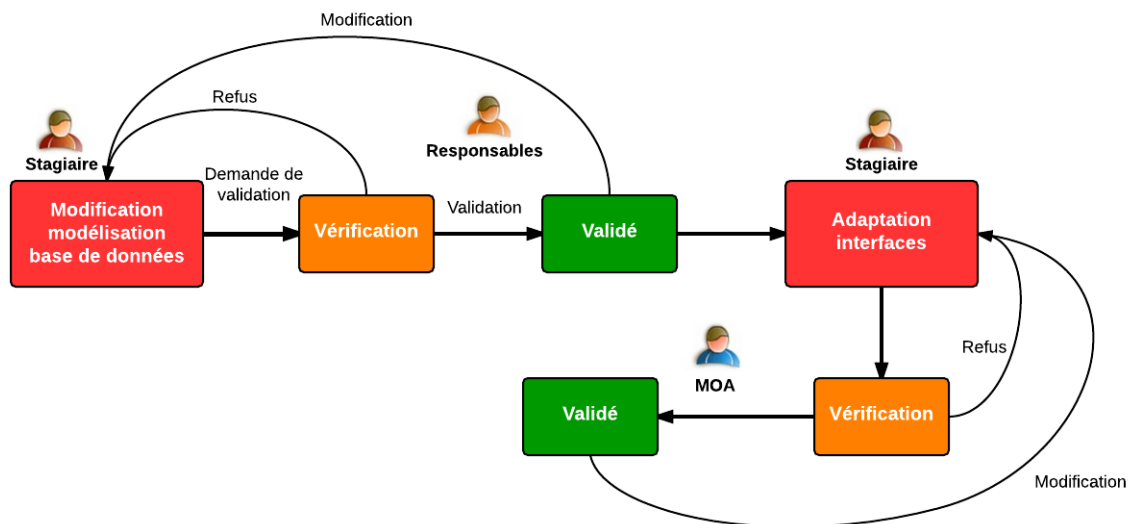


FIGURE 4.6 – Worflow du projet

Chapitre 5

Les solutions techniques

Au cours de ce chapitre, je vais vous présenter les différentes solutions techniques mises en oeuvre pour la réalisation du projet. Des décisions techniques ont été prises tout au long de la réalisation du projet. C'est pourquoi, j'ai décidé de présenter chaque étape de réalisation et de détailler pour chacune les solutions ayant été choisies.

5.1 Synthèses de l'existant

Avant de vous décrire la partie technique de mon travail, je vais vous introduire les principales technologies et principaux logiciels que j'ai utilisés.

5.1.1 Langage de développement et concepts de programmation

J2EE

L'application sur laquelle je travaille est développée en Java J2EE, soit Java Entreprise Edition. C'est la version de Java destinée aux applications des entreprises. J2EE constitue une collection de composants, de conteneurs et de services permettant de créer et de déployer des applications distribuées au sein d'une architecture standardisée. L'un des intérêts d'une programmation objet est de construire des objets indépendants les uns des autres, pour pouvoir les réutiliser par la suite, dans d'autres programmes.

Architecture 3 tiers J2EE se fonde sur un modèle en plusieurs couches, on parle d'architecture multi-tiers : tier client, tier serveur J2EE composé de tier web et de tier métier puis le tier EIS (Entreprise Information Systems).

Le tier client est un client lourd car Amex est basé sur Spring.

Le tier métier utilise des EJB (Entreprise JavaBean), notamment les Session Bean qui implantent un dialogue client serveur pour accomplir une tâche.

Le tier EIS concerne la base de données (Sybase et SQL), il est matérialisé par des classes Java au niveau tier métier.

image application 3 tiers

Le design pattern IoC

En temps normal, une application définit ses propres variables et objets. Elle se construit de manière autonome. Cela peut être gênant lorsque certains éléments ne sont pas connus à l'avance, et ne seront connus que lors de l'exécution du programme. Pour cette raison, le design pattern IoC, ou Inversion Of Control a été mis en place.

L'inversion de contrôle est un patron d'architecture commun à tous les frameworks (ou cadre de développement et d'exécution). Il fonctionne selon le principe que le flot d'exécution d'un logiciel n'est plus sous le contrôle direct de l'application elle-même mais du framework ou de la couche logicielle sous-jacente. L'inversion de contrôle est un terme générique. Selon la problématique, il existe différentes formes, ou représentations d'IoC. La plus connue étant l'injection de dépendances qui est un patron de conception permettant, en programmation orientée objet, de découpler les dépendances entre objets.

Dans notre cas c'est Spring qui se charge de mettre en relation les objets entre eux. Ainsi, dans le cas d'un programme nécessitant de se connecter à un serveur, mais dont l'adresse du serveur peut changer, un programme classique obligerait de modifier le code source à chaque changement d'adresse du serveur. Spring permet de créer un fichier séparé contenant ces informations, et de lier automatiquement la valeur de ce fichier à la variable contenue dans le programme utilisant cette valeur.

5.1.2 Les outils utilisés

Sybase pour la base de données.
Eclipse puis IntelliJ comme IDE de travail.
SVN puis GIT comme outil de gestion de version.

Hibernate
Spring

Certains termes techniques seront utilisés au long de cette section. En annexe, vous trouverez une partie lexicale définissant les termes suivis d'une *.

5.2 Description du travail

5.2.1 Analyse de l'existant

Le diagramme entité relation représentant l'ancienne modélisation est en annexe :

Les API utilisant les FlowDetails

Dans la plateforme Amex, il y a différentes services qui utilisent les flowdetails. Je vais décrire le fonctionnement de chacun d'eux.

Le service d'audit a pour but de tracer tous les messages transitant par Amex. L'audit se fait dans des tables en base de données. Il existe différents statuts d'Audit décrivant le statut du message transitant, par exemple un message a comme statut PENDING à sa génération, SENT à son envoi si un acquittement est attendu, SENT NO_ACK à son envoi si aucun acquittement est attendu, ACKNOWLEDGED à son acquittement, ERROR en cas d'erreur .

Il est possible de paramétrer des taches automatiques à exécuter lorsqu'un message transitant par Amex passe dans un certain statut d'audit. C'est le role du service de taches automatiques. Ces taches automatiques peuvent etre un envoi d'un mail, un appel à un web service distant ou le post d'un message sur une file JMS ou MQ. Il existe des objets TaskHandler permettant de savoir comment créer le message à envoyer et à qui l'envoyer. Ces informations sont contenues dans la table TaskHandler et sont paramétrables via des arbres RuleSolver.

5.2.2 La nouvelle modélisation

La première étape a été de réaliser une nouvelle modélisation base de données des composants en relation avec les FlowDetails. Cette modélisation doit être évolutive et cohérente avec le besoin des MOA.

Le diagramme entité relation a été réalisé avec le logiciel VPUml et se trouve en annexe ... lien

De nombreuses modifications ont été réalisées, je vais seulement décrire les plus importantes.

Une FlowDetail décrit l'entrée ou la sortie d'un flux et contient des informations nécessaires pour l'audit des messages mais aussi pour son routage. Pour l'envoi d'un message il est nécessaire de connaître son média de sorti, son format de sorti mais aussi sa destination. Il existe 7 types de média différents :

1. PRINTER
2. FILE
3. JMS
4. FAX
5. WEB SERVICE
6. MQ
7. MAIL

Les informations sur la destination d'un message dépendent du média utilisé, par exemple un média FAX est représenté par un numéro de FAX, un média MAIL par un ensemble d'adresses mail...etc Un média MAIL est donc un ensemble d'adresses mail accompagnés de leur catégorie (mail bb, mail cc...).

Il existe une table différente pour chaque type de média, ce qui permet de typé fortement les médias, c'est à dire qu'un média peut contenir différentes informations qui lui sont propres. Par exemple pour envoyer un message par média FAX, seul le numéro de fax est utile, par contre pour envoyer un message par média JMS, il faut connaître le nom de la file JMS ainsi que le serveur l'hébergeant. Cette modélisation permet donc de modifier très rapidement un média si cela est nécessaire.

Un message transitant dans Amex peut avoir différents format.

Lorsqu'un message passe dans un certain statut d'audit, il est possible de paramétrer des tâches à exécuter, comme l'envoi d'un email ou l'appel à un web service distant ou autre tâche. Afin de savoir comment créer le message, la méthode d'une classe java est appelée. Ces informations étaient enregistrées en bdd dans la table TASK_HANDLER. Mais les informations dont on a besoin ici correspondent exactement à celles enregistrées dans l'objet flowdetail à la seule différence prêt que le message est créé avec l'appel d'une méthode java. C'est pourquoi il existe un type de format custom qui est utilisé pour ce service. Ainsi le nom du format sera le nom de la transformation à utiliser, le sous-format sera le nom de la classe à appeler et le type "custom". Il est important de pouvoir garder une trace de toutes les actions effectuées. Nous proposons donc un système d'historique pour les flowdetails, les médias et les contacts. Lorsqu'un FlowDetail par exemple est modifié (changement de label ou erreur dans sa valeur), celui-ci n'est pas supprimé mais devient inactif. Un FlowDetail actif a une date de validité de fin valant "9999-12-31". L'exemple ci-dessous montre le fonctionnement :

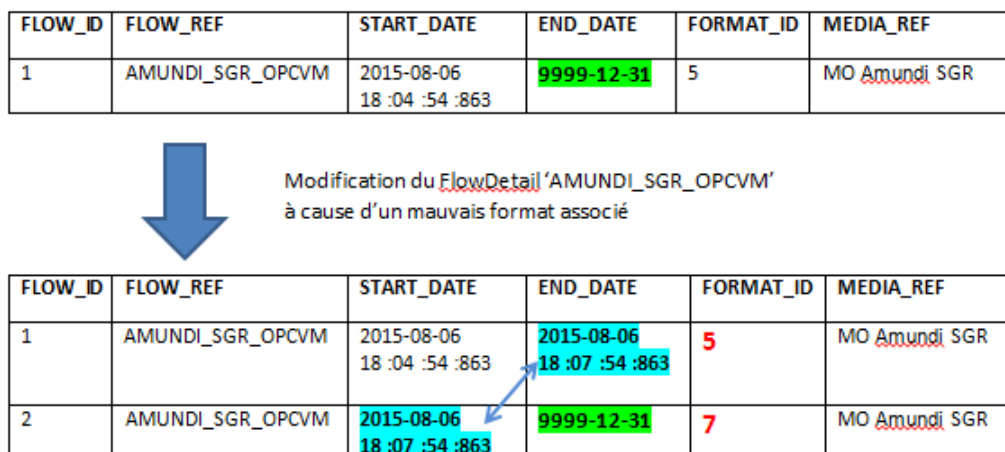


FIGURE 5.1 – Modification d'un FlowDetail

C'est la référence du média qui est stocké dans la table FlowDetail. Ainsi si un média est modifié, il n'est pas utile de modifier les flowdetail utilisant ce média, car sa référence reste la même mais son id change . Pour connaître le média associé à un FD, une jointure avec les tables de médias est effectuée, et c'est le média avec la plus grande date de validité de fin qui est sélectionné. Cela explique pourquoi certains liens n'apparaissent pas dans la modélisation.

5.2.3 Migration de l'existant

Amex contient déjà un certains nombres de flux et donc de nombreux FlowDetails sont déjà sauvegardés en base de donnée. Il a donc fallu migrer les données de l'ancienne modélisation vers la nouvelle sans en perdre une seule. J'ai décidé de faire ce programme de migration en JAVA et de recopier les données "tables par tables". L'avantage d'utiliser Java était notamment de pouvoir réaliser facilement des tests afin d'être sur que rien n'avait été perdu. Tout d'abord les schedules ont été recopiés, puis les formats, parties, les médias, les contacts et les FlowDetails. Les arbres utilisant les TaskHandler ont du être modifiés.

La nouvelle modélisation contient uniquement des noms de table différents. Le programme consiste donc à peupler les nouvelles tables avec les données existantes.

La seule modification devant avoir lieu directement sur une table existante concerne les arbres RuleSolver utilisant des TaskHandler. L'id du TaskHandler était stocké en base de données afin de retrouver le TaskHandler correspondant à un noeud résultat. Il est plus générique et plus correcte de stocker une référence au lieu d'un id, ainsi lors de la modification d'une modification, la référence reste la même donc aucune modification ne doit être effectuée sur les arbres. C'est donc la référence du FlowDetail (celui correspondant à l'ancien TaskHandler) qui est stockée dans ces arbres RS.

Pour un message SWIFT, seul était enregistré les coordonnées du contact swift sender

et receiver, mais en réalité ce message SWIFT est tout d'abord envoyé par file MQ par exemple puis sur la Plateforme Swift. Les noms de files MQ correspondantes étaient stockées dans un fichier properties directement dans le projet. De plus un fichier xml permettait de trouver la bonne MQ en fonction du BIC du sender Swift et du Receiver Swift :

```
<bean id="SwiftGsmSgss54XCriteria"
class="com.sits.frameworkstp.core.messaging.router.swift.SwiftRoutingCriteria">
  <property name="swiftMessageType" value="MT54X" />
  <property name="senderBicCodePrefix" value="AGRIFRP" />
  <property name="senderBicCodeSuffix" value="GSM" />
  <property name="receiverBicCodePrefix" value="SOGEFRP" />
  <property name="swiftChannel" value="SwiftGsmSgss54XChannel" />
</bean>
```

FIGURE 5.2 – Routage d'un FlowDetail Swift

Lors de l'appel au bean `SwiftGsmSgss54XCriteria`, une méthode la classe `SwiftRoutingCriteria` est appelée et permet de trouver le bon channel en fonction des BIC du sender et receiver. Ainsi un si les BICS correspondent à ceux de la figure ci dessus, alors le channel correspondant est `SwiftGsmSgss54XChannel`, ce dernier permettra de retrouver le nom de la file MQ.

Avec la nouvelle modélisation, ces étapes ne sont plus nécessaire car ces données seront directement stockées en base.

Afin de connaître la valeur des médias correspondants au FlowDetail SWIFT, il a donc fallu parcourir les fichiers XML puis les fichiers properties. Le parcourt des fichiers XML a été fait à l'aide d'un outil appelé JDOM dont le but est de manipuler un document Xml. Pour chaque bean ayant la même structure que celui ci-dessus, une ligne était créé et enregistré dans un nouveau fichier properties :

```
.MT54X.AGRIFRP.GSM.SOGEFRP=StpSwiftGsmSgss54XQueue
```

FIGURE 5.3 – Ligne correspondante créée

Dans le programme de migration, la clé correspondante est créé en fonction du swift sender et swift receiver, il est ainsi possible de récupérer sa valeur (le channel) dans le nouveau fichier properties. La valeur du média est ensuite récupérée dans l'autre fichier properties ou les nom de MQ sont définis et associées aux channels.

5.2.4 Refonte des IHM

Un fois la modélisation validée et les tables peuplées de données, les services ont été mis en place. L'interface a été conçue selon le design pattern MVC vu précédemment.

```
Sgss.nantes.54x.in.queue=StpSwiftGsm54XQueue
Sgss.nantes.54x.out.queue=QA.ESB.SGSSNANTES.ETD
```

FIGURE 5.4 – Ancien fichier properties

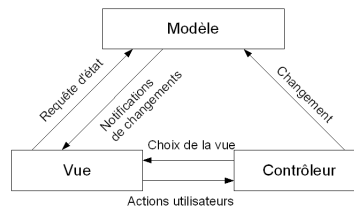


FIGURE 5.5 – Ancien fichier properties

Le modèle correspond ici aux objets métier et aux informations de la base de données. Il contient donc les données qui seront affichées par la vue et sont contenus dans le projet flowdetails-common ainsi

La vue correspond aux interfaces avec lesquelles l'utilisateur peut interagir. Plusieurs modules du logiciel MediaPlus Alto ont été modifiés : FlowDetails Viewer, les modules des arbres RuleSolver et STP-Monitoring. La principale fenêtre est FlowDetails Viwer et permet aux utilisateurs de manipuler les FlowDetails.

Le controleur fait le lien entre la vue et le modèle. Il reçoit des éléments transmis par la vue et exécute les actions correspondantes.

Utilisation de JIdeSoft Travail sur l'IHM : trop d'étapes pour créer un FD, trop de validation et de fenêtres qui s'ouvrent

5.2.5 Bascule sur les nouveaux services

5.3 Protocole d'évaluation

Chapitre 6

Avancement du projet

Chapitre 7

Impressions personnelles

Chapitre 8

annexe

8.1 Lexique

Flux Un flux permet l'envoi et la reception de données. Ils traitent les données de manière séquentielle ou asynchrones.

Swift

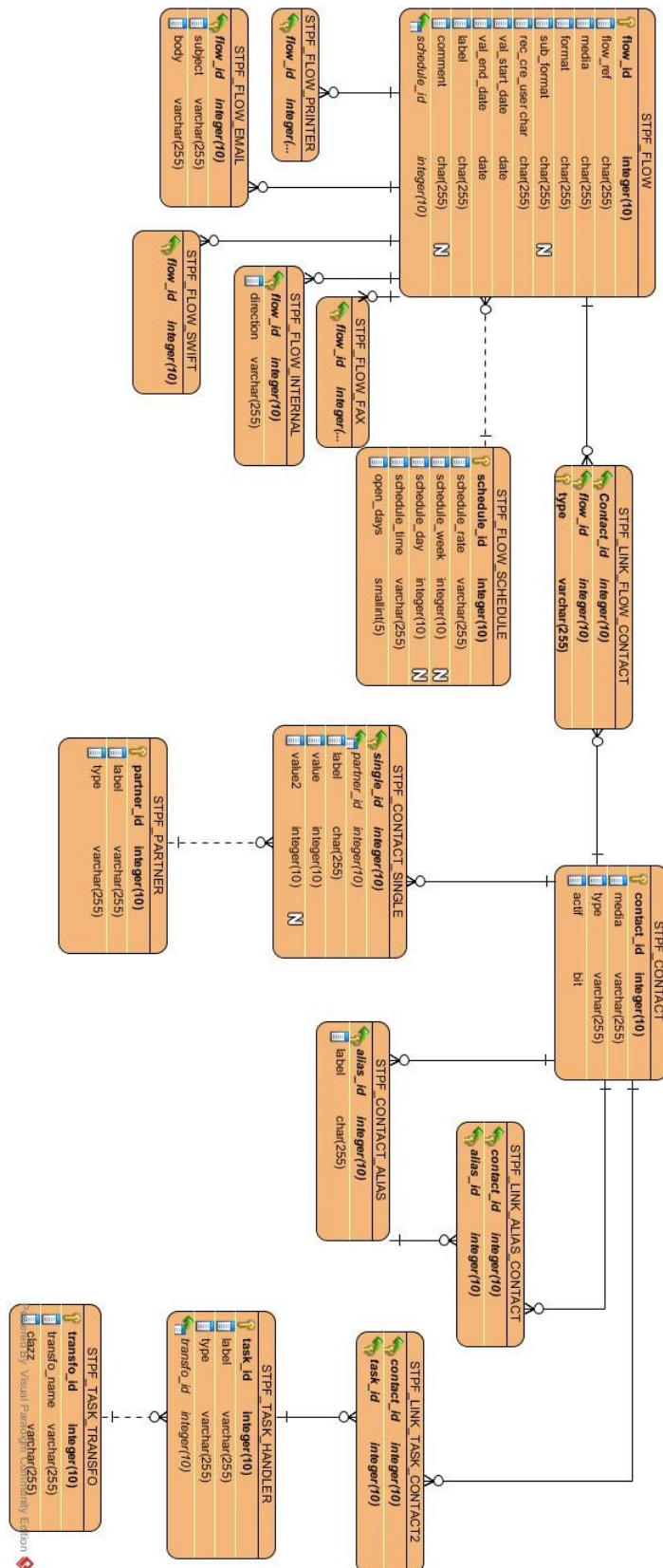


FIGURE 8.1 – Diagramme entit  relation de la mod lisation existante

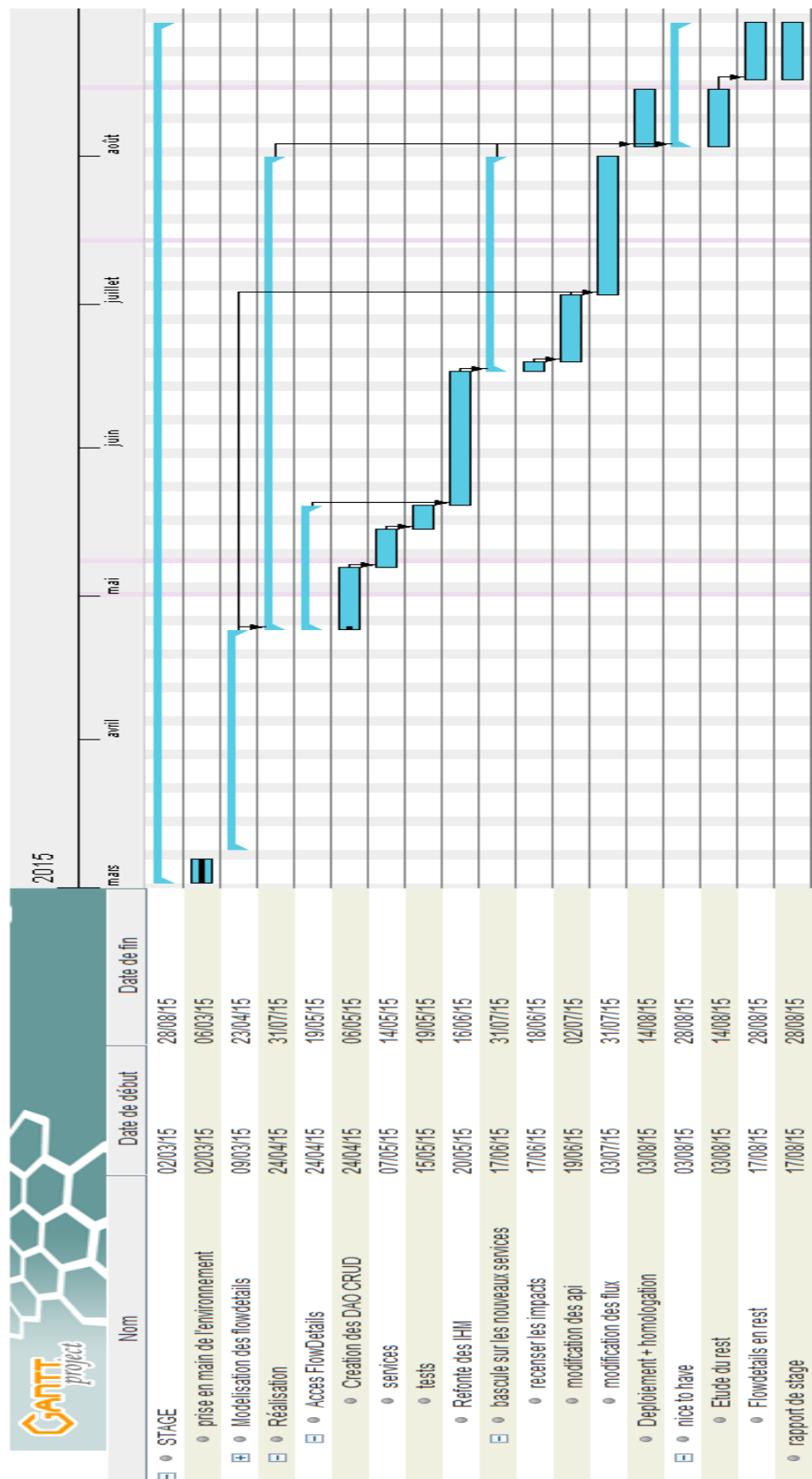


FIGURE 8.2 – Diagramme de Gantt