Sodifrance
IT transformation to digital

Rapport de stage

CONSEIL, SOLUTIONS DE TRANSFORMATION ET SERVICES IT

1. PRESENTATION DE L'ENTREPRISE

1.1 CHIFFRES CLES

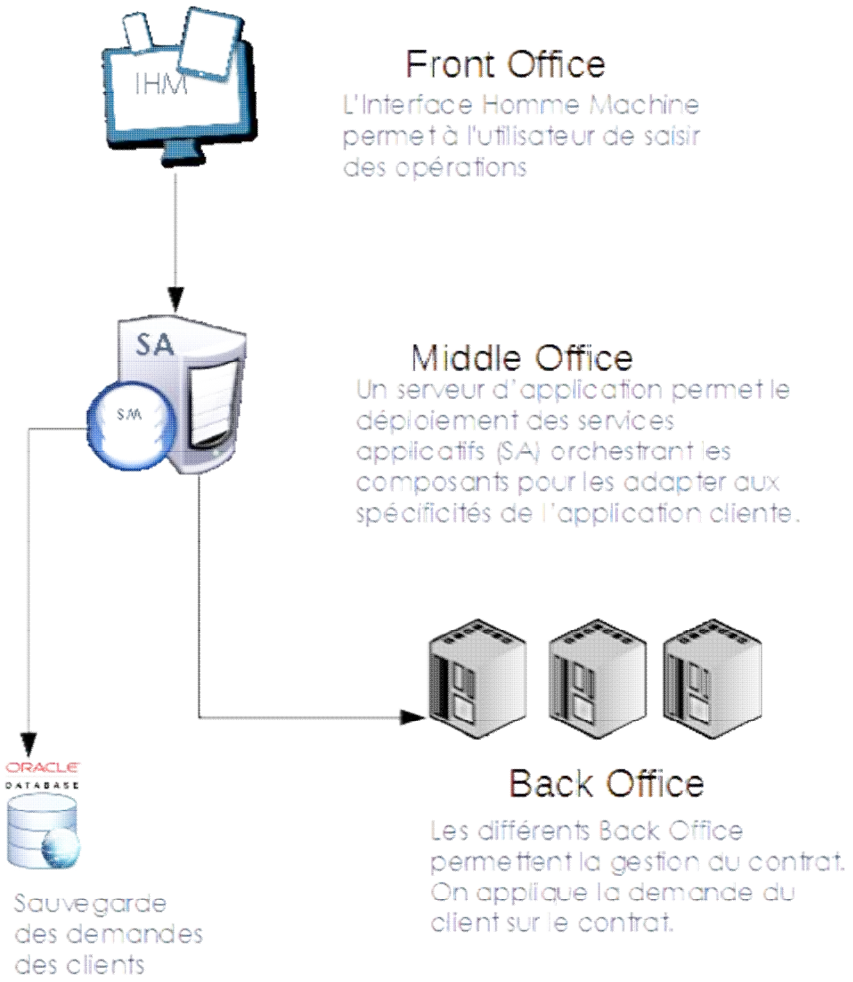
Sodifrance est une SSII française fondée en 1986. Elle compte 906 collaborateurs répartis en 13 implantations en France et Tunisie. En 2013 le CA est de 71,4M€.

1.2 VALEURS DE L'ENTREPRISE :

Une entreprise de proximité évoluant sur un plan national.
Une capacité à accompagner ses clients sur des projets stratégiques.
Une culture technique et le respect des engagements.

1.3 LE CENTRE DE SERVICES D'ANGERS

Le pôle angevin d'une vingtaine de personnes travaille principalement pour le client CNP (Caisse Nationale de Prévoyance). Son rôle est de réaliser des composants logiciels développés en langage Java. D'un point de vue général Sodifrance intervient sur des composants de la couche Middle Office, c'est-à-dire que les composants centraux du service d'informations de la CNP qui permettent l'interaction entre le Front Office (la couche présentation l'IHM) et la couche Back Office qui a la responsabilité de la gestion des différents contrats d'assurance vie d'un assuré. Les composants sur lesquels travaillent Sodifrance ont donc une forte connotation métiers puisqu'ils doivent garantir la validité des données qui s'échangent entre le Front Office et le Back Office. Le schéma suivant explicite ce principe :


IHM
Front Office
L'Interface Homme Machine permet à l'utilisateur de saisir des opérations
SA
SM
Middle Office
Un serveur d'application permet le déploiement des services applicatifs (SA) orchestrant les composants pour les adapter aux spécificités de l'application cliente.
ORACLE
DATABASE
Sauvegarde des demandes des clients
Back Office
Les différents Back Office permettent la gestion du contrat. On applique la demande du client sur le contrat.

2. MISSION

Mise en place d'un outil en langage Java permettant de générer une vision graphique des échanges entre composants (de type diagramme de séquences) à partir des flux tracés lors de la saisie d'un processus à partir d'une application cliente.

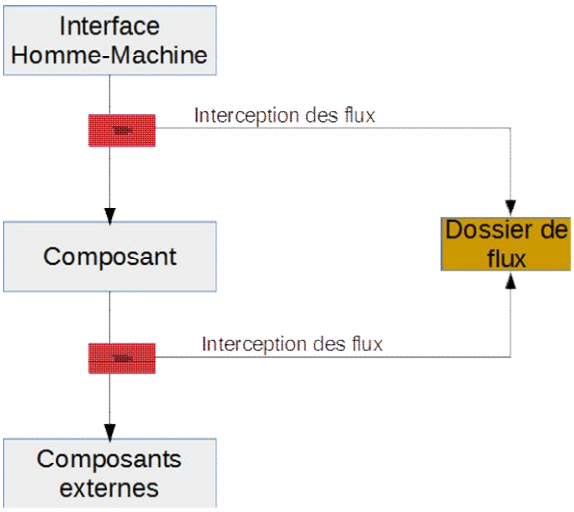
2.1 PROBLEMATIQUE

Sodifrance livre des composants logiciels à la CNP, ces composants logiciels sont livrés (sous forme de .jar) puis sont intégrés au sein des différentes applications clientes qui les utilisent.

Une fois les composants livrés intégrés, la CNP planifie des phases de qualifications logicielles afin de vérifier la conformité des composants livrés avec les spécifications métiers attendues.

Lors d'une phase de tests, l'ensemble des données échangées entre les différents composants Front Office, Middle Office et Back Office sont sauvegardés sous forme de flux XML et ce notamment afin de :

- Faciliter le diagnostic lors d'une anomalie par l'analyse des données tracées
- Permettre de rejouer un scénario de tests à partir des données tracées pour notamment reproduire l'anomalie et vérifier qu'elle a bien été corrigée



Cependant pour le moment les différentes traces récupérées ne sont exploitables que par des collaborateurs qui connaissent le fonctionnement des différentes interactions entre les composants.
Les personnes qui réalisent les tests possèdent uniquement une vision métier et fonctionnelle – ils saisissent des cas de tests via une IHM et vérifient le résultat de leur saisie une fois les données envoyées sur le Back Office.
Ils n'ont donc pas connaissance de tous les composants sollicités lors d'un test.
Lorsqu'une anomalie est constatée il est donc difficile pour ces équipes de tests d'affecter l'anomalie au bon composant responsable de sa correction.

2.2 OBJECTIFS

L'outil à mettre en place doit donc permettre de donner une vision graphique des différents échanges entre les composants sollicités lors de la saisie d'un scénario de tests afin de faciliter la compréhension des interactions entre ces différents composants.
La restitution graphique devra être effectuée sous forme de diagramme de séquences afin de permettre également la reprise de ces diagrammes de séquences notamment pour les processus métiers non documentés.
On utilise ici la connotation didacticielle des tests, on part des tests pour expliciter les échanges entre les différents composants et ainsi faciliter la compréhension du processus métier.

3. DEVELOPPEMENT DE L'OUTIL

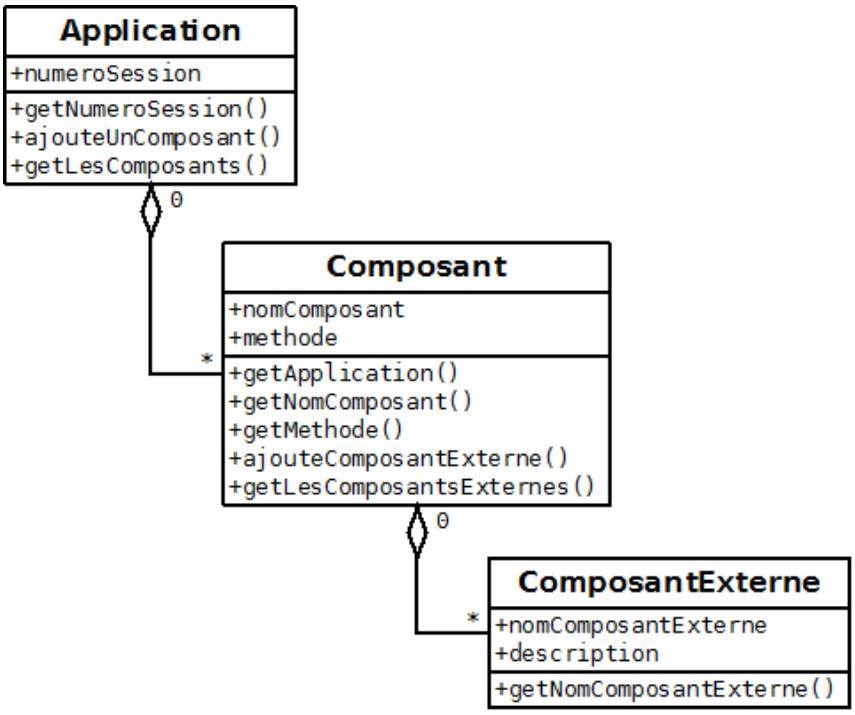
3.1 PLANNING ASSOCIE

Semaine 1 – Mise en place du framework technique
Semaine 2 – Choix de la solution pour l'affichage graphique du diagramme de séquence
Semaine 3 - Réalisation
Semaine 4 et 5 - Intégration avec les outils existants

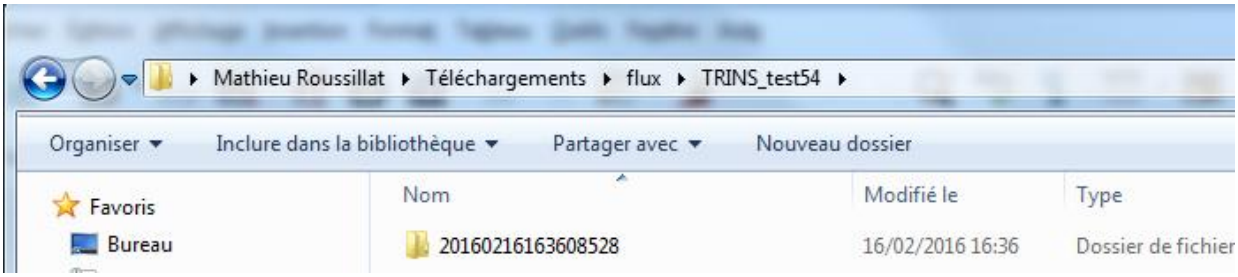
3.2 MISE EN PLACE DE FRAMEWORK TECHNIQUE

Tout d'abord j'ai dû réaliser l'installation de mon environnement de travail : Eclipse, JDK.
Ensuite j'ai téléchargé des dossiers de flux que mon maître de stage m'a envoyé afin de pouvoir tester mon programme lors du développement.

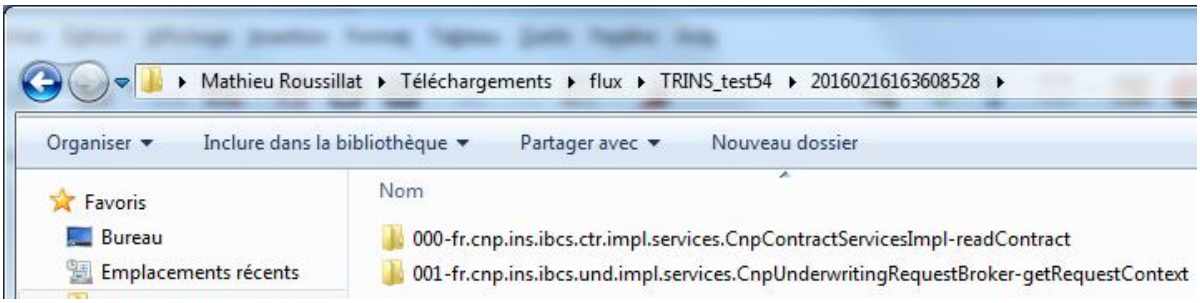
Puis il a fallu définir le modèle objets pour la construction graphique :



Voici un exemple de contenu d'un dossier de flux que mon maître de stage m'a envoyé. Dans cet exemple le programme doit créer un objet Application avec pour numéro de session 20160216163608528. Le nom du dossier correspond au numéro de la session.

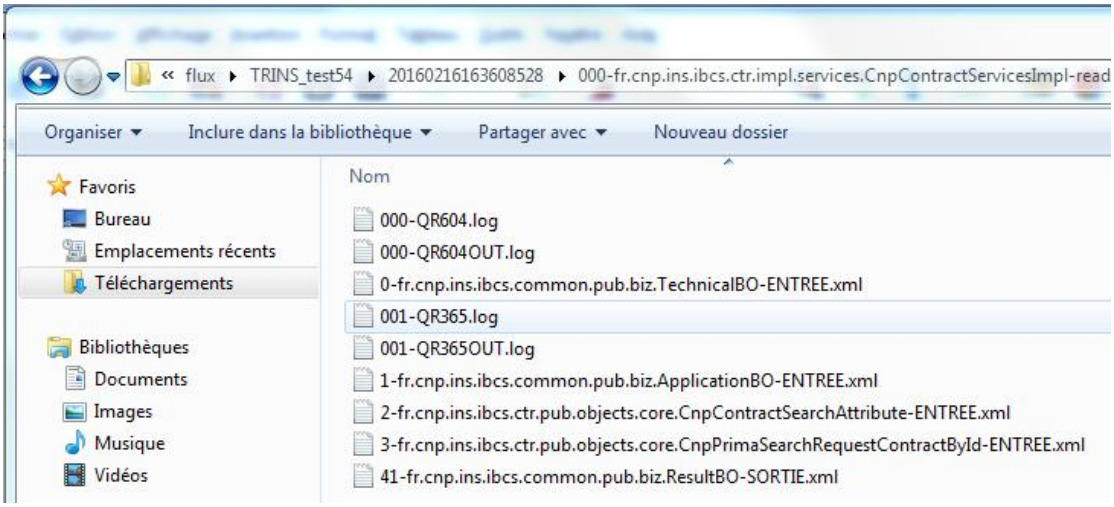


Le programme doit créer un objet Composant pour chaque dossier de composant sollicité avec le nom de la méthode utilisée.



Le nom du composant est situé après -fr.cnp et les méthodes sont situées après le dernier séparateur - .
Ici le composant INS est sollicité 2 fois, la première fois avec la méthode readContract et la deuxième fois avec la méthode getRequestContext.

Le programme doit faire la liste des données Entrée/Sortie des composants externes sollicités qui sont sauvegardées dans des fichiers de type QR*.log. Pour chacun de ces fichiers il doit créer un objet ComposantExterne.



Le programme à écrire doit permettre de retracer ces flux en parcourant récursivement les dossiers de flux et afficher le résultat sous forme de diagramme de séquence.

J'ai donc commencé l'écriture du programme en suivant le modèle objet défini plus tôt.

Voici le résultat produit par le programme pour cet exemple :

```
Le numéro de session est le : 20160216163608528
    Le nom du composant sollicité est ins
    Le nom de la méthode utilisée est readContract
        Le nom du composant externe utilisé est QR604.log
        Le nom du composant externe utilisé est QR604OUT.log
        Le nom du composant externe utilisé est QR365.log
        Le nom du composant externe utilisé est QR365OUT.log
    Le nom du composant sollicité est ins
    Le nom de la méthode utilisée est getRequestContext
        Le nom du composant externe utilisé est QR604.log
        Le nom du composant externe utilisé est QR604OUT.log
        Le nom du composant externe utilisé est QR365.log
        Le nom du composant externe utilisé est QR365OUT.log
        Le nom du composant externe utilisé est QR365.log
        Le nom du composant externe utilisé est QR365OUT.log
        Le nom du composant externe utilisé est QRM37.log
        Le nom du composant externe utilisé est QRM37OUT.log
        Le nom du composant externe utilisé est QRM37.log
        Le nom du composant externe utilisé est QRM37OUT.log
```

Liste des compétences :

A1.1.1 Analyse du cahier des charges d'un service à produire
- C1.1.1.1 Recenser et caractériser les contextes d'utilisation, les processus et les acteurs sur lesquels le service à produire aura un impact
- C1.1.1.2 Identifier les fonctionnalités attendues du service à produire

➔ Modèle objet.

A1.1.2 Étude de l'impact de l'intégration d'un service sur le système informatique
- C1.1.2.1 Analyser les interactions entre services
- C1.1.2.2 Recenser les composants de l'architecture technique sur lesquels le service à produire aura un impact

➔ Modèle objet.

A4.1.6 Gestion d'environnements de développement et de test

C4.1.6.1 Mettre en place et exploiter un environnement de développement

➔ Installation de mon environnement de travail)

A4.1.7 Développement, utilisation ou adaptation de composants logiciels

C4.1.7.2 Créer un composant logiciel

➔ Création du programme

A5.2.3 Repérage des compléments de formation ou d'auto-formation utiles à l'acquisition de nouvelles compétences

C5.2.3.1 Identifier les besoins de formation pour mettre en oeuvre une technologie, un composant, un outil ou une méthode

C5.2.3.2 Repérer l'offre et les dispositifs de formation

➔ lecture javadoc et tutos sur les classes File et String pour utiliser les méthodes listfiles(), getName(), getParentFile(), split().

A5.2.4 , Étude d'une technologie, d'un composant, d'un outil ou d'une méthode.

C5.2.4.1 Se documenter à propos d'une technologie, d'un composant, d'un outil ou d'une méthode.

➔ lecture javadoc et tutos sur les classes File et String pour utiliser les méthodes listfiles(), getName(), getParentFile(), split().

3.3 CHOIX DE LA SOLUTION POUR L'AFFICHAGE GRAPHIQUE DU DIAGRAMME DE SEQUENCE

Comme une fois fini le programme doit afficher un diagramme de séquence, j'ai dû chercher et tester différents plugins et quelques bibliothèques permettant de créer des diagrammes UML.

Voici les plugins que j'ai testés :

Nom	UML Designer	ObjectAid UML Explorer	Umlet	UML Lab
Avantages		très simple, faire glisser les classes et méthodes vers le diagramme pour créer des objets.		
Inconvénients	Pas de diagramme à partir de code	Diagramme de séquence payant	Pas de diagramme de séquence automatique	Cher
Prix	Gratuit	Gratuit Certaines options sont payantes	Gratuit	Payant : 199€

Nom	Sirius	Graphical Editing Framework (Zest, Draw2d)	Papyrus	Modisco	ModelGoon
Avantages					Assez simple
Inconvénients	Pas de diagramme à partir de code		Très lourd et compliqué	Pas de diagramme à partir de code	
Prix	Gratuit	Gratuit	Gratuit	Gratuit	Gratuit

Et voici les bibliothèques que j'ai essayées :

Nom	JgraphT/Swing	PlantUML	JfreeChart
Avantages	Assez simple	Très simple et spécialisé dans les diagrammes UML	
Inconvénients	N'est pas spécialisé dans les diagrammes UML	Obligation de décrire les interactions dans un langage compréhensible pour la bibliothèque.	Ne fait pas de diagrammes de séquence.
Prix	Gratuit	Gratuit	Gratuit

La bibliothèque PlantUML, contrairement à JgraphT, est spécialisée dans les diagrammes UML.
Elle est aussi beaucoup plus simple à utiliser que les plugins ModelGoon et Umlet.
Cette solution est gratuite.

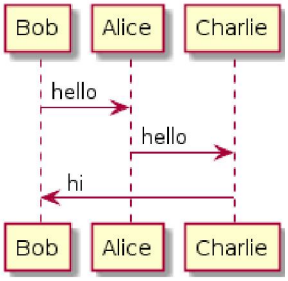
Donc la solution retenue est **PlantUML**.

Fonctionnement de PlantUML :

La bibliothèque PlantUML reçoit ce texte :

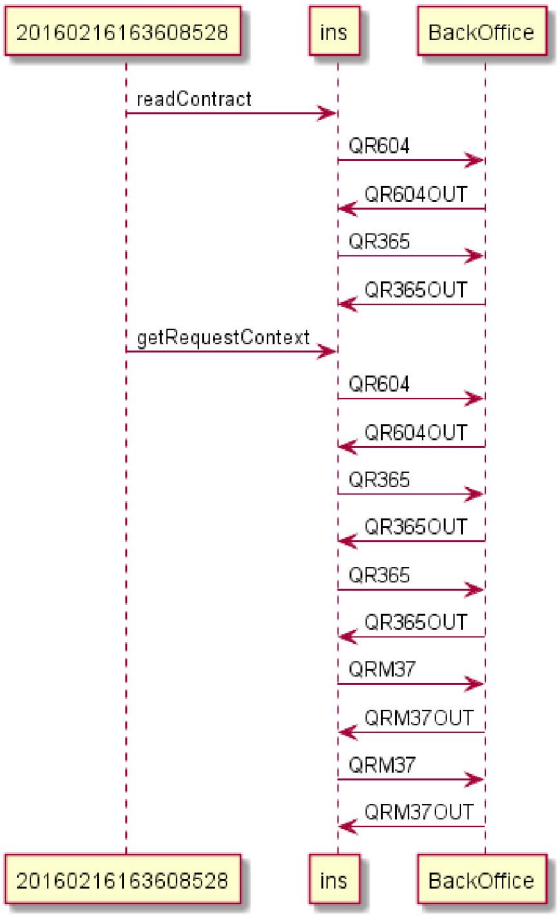
```
Bob->Alice : hello
Alice->Charlie : hello
Charlie->Bob : hi
```

Et le transforme en ce diagramme de séquence :



Le programme devra envoyer les informations à la bibliothèque PlantUML sous la forme « Composant → Composant externe : nomDeLaMethode » par exemple.
PlantUML produira un diagramme de séquence à partir de ces informations et l'enregistrera dans un format d'image.

Voici l'image retournée par le programme pour l'exemple partie 3.2 après l'ajout de PlantUML :



Liste des compétences :

A1.2.1 Élaboration et présentation d'un dossier de choix de solution technique

C1.2.1.1 Recenser et caractériser des solutions répondant au cahier des charges (adaptation d'une solution existante ou réalisation d'une nouvelle).

C1.2.1.2 Estimer le coût d'une solution

C1.2.1.3 Rédiger un dossier de choix et un argumentaire technique.

➔ Choix de la solution pour générer les diagrammes de séquence.

A4.1.7 Développement, utilisation ou adaptation de composants logiciels

C4.1.7.2 Créer un composant logiciel

➔ Modification du programme

A5.2.3 Repérage des compléments de formation ou d'auto-formation utiles à l'acquisition de nouvelles compétences

C5.2.3.1 Identifier les besoins de formation pour mettre en oeuvre une technologie, un composant, un outil ou une méthode

C5.2.3.2 Repérer l'offre et les dispositifs de formation

➔ lecture de tutos sur la bibliothèque PlantUML.

A5.2.4 , Étude d'une technologie, d'un composant, d'un outil ou d'une méthode.

C5.2.4.1 Se documenter à propos d'une technologie, d'un composant, d'un outil ou d'une méthode.

➔ lecture tutos sur la bibliothèque PlantUML.

3.4 REALISATION

Le programme a été réalisé en plusieurs parties.
Au début il faisait du parcours récursif de dossiers. Il déterminait l'application puis les composants avec les méthodes sollicitées et enfin il a pu faire la liste des composants externes utilisés.

Après, la possibilité de réaliser un diagramme de séquences a été ajoutée au programme.
Cette partie a été assez facile à réaliser, la transformation des informations reçues par le programme vers le langage PlantUML étant très simple.

Ensuite il a fallu améliorer le programme afin de pouvoir choisir le dossier de flux à parcourir et modifier le mode d'affichage du diagramme de séquences. Cette partie a permis de découvrir des problèmes non rencontrés dans les parties précédentes.

3.4.1.1. MODIFICATION DU MODE DE SELECTION DU DOSSIER DE FLUX A PARCOURIR

Lorsque l'utilisateur veut faire parcourir un dossier de flux au programme, il doit modifier une variable dans le code du programme.
De plus un antislash \ seul n'est pas considéré comme un caractère valide. Il doit être doublé ou remplacé par un slash / . Donc un copier-coller n'est pas suffisant. Il faut ensuite modifier les antislash \ dans l'adresse.

Pour tester le programme 4 dossiers de flux de composition différente étaient utilisés.
Certains contenaient plusieurs sessions, d'autre plusieurs composants sollicités ou encore pas de composant externe appelé.

La variable de nom de dossier de flux à parcourir était présente 4 fois dans le programme dont 3 étaient mises en commentaire et n'étaient pas utilisées pendant l'exécution du programme.
Pour changer de dossier de flux il suffisait de dé commenter la ligne contenant le dossier à explorer et commenter les autres.

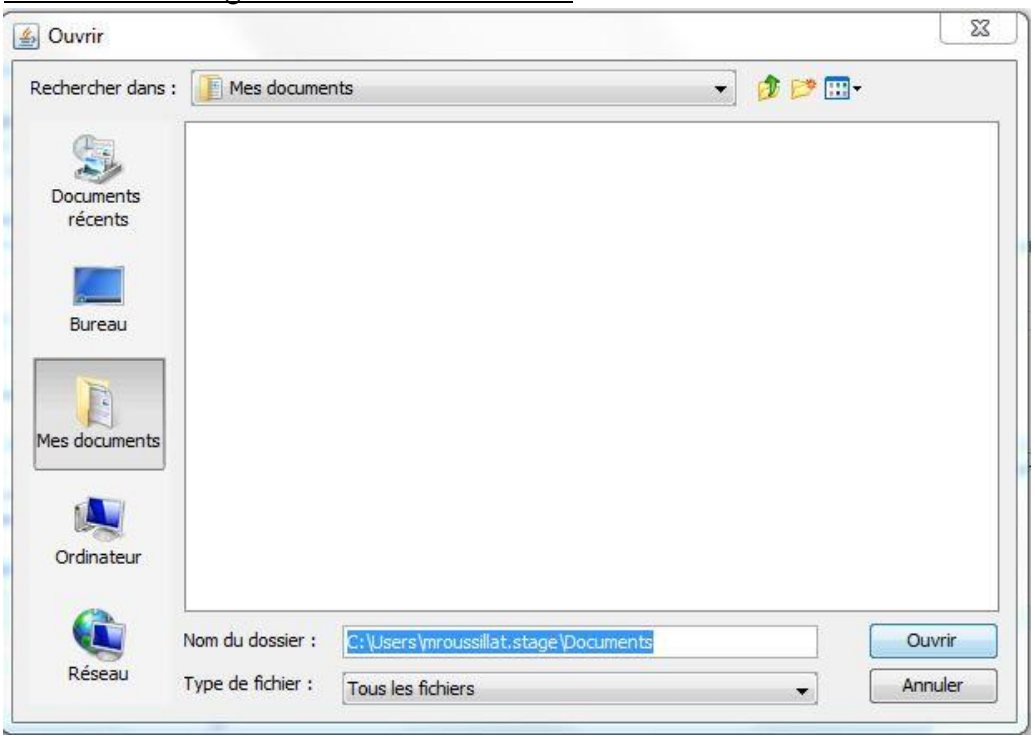
```
// on donne 1 dossier
//String nom = "C:/Users/mroussillat.stage/Downloads/flux/TRINS_test53ter";
String nom = "C:/Users/mroussillat.stage/Downloads/flux/TRINS_test54";
//String nom = "C:/Users/mroussillat.stage/Downloads/Flux dézippés/TRINS_test56";
//String nom = "C:/Users/mroussillat.stage/Downloads/Flux dézippés/TRINS_test58";
```

A la base il n'y avait qu'un seul dossier de flux puisqu'il fallait déjà que le programme fonctionne pour un dossier avant d'en tester d'autres. Puis en avançant dans le développement il a fallu tester le programme avec des dossiers de flux plus complexes et les variables portant le même nom se sont rajoutées.
Un tableau aurait pu être utilisé au lieu d'utiliser le système de commenter/dé commenter des lignes mais le problème de modification du code aurait été le même.

Comme il n'est pas aisé de changer de dossier de flux et que si l'on veut en tester un nouveau il faut le rajouter dans le code du programme il faut permettre à l'utilisateur de choisir quel dossier de flux parcourir autrement qu'en modifiant le code du programme.

Résultat :
L'utilisateur peut maintenant choisir le dossier de flux à parcourir grâce à une boîte de dialogue de sélection de fichiers. Il n'a donc plus à modifier le code du programme pour changer de dossier.
Cette boîte de dialogue est créée grâce au composant JFileChooser de la bibliothèque Swing. Ce composant crée lui-même la boîte de dialogue.

La boîte de dialogue lors de son ouverture :



Cette partie a permis de régler un problème crée dans la première partie de la création du programme.

3.4.1.2. AFFICHAGE DU DIAGRAMME A L'ECRAN

Pour l'instant le programme génère un diagramme de séquence à partir du dossier de flux sélectionné et l'enregistre sous forme d'image au format .png sur le disque dur à un emplacement défini dans le code du programme.
PlantUML propose deux formats de sortie :

- .png
- .svg

C'est le format .png qui a été choisi car il n'y a pas besoin d'installer de logiciel spécial pour ouvrir ce format d'image.

Mais pour savoir où retrouver l'image l'utilisateur doit regarder dans le code du programme.
Pour éviter cela le programme doit afficher le diagramme de séquence automatiquement à l'écran.

Cette partie a été la plus riche en problèmes.

a) Problème 1 :

Après plusieurs tests, un problème est apparu. Certains diagrammes ont une hauteur qui dépasse la capacité d'affichage de l'écran d'ordinateur. Le diagramme est donc coupé.

Dans les captures d'écran de ce problème les fenêtres ont été redimensionnées et la barre des tâches de Windows a été réglée pour être masquée automatiquement afin d'avoir la plus grande taille d'image possible pour cette comparaison.

Pour ce cas le diagramme est coupé un peu avant la moitié.
Il faut donc ajouter des barres de défilement.

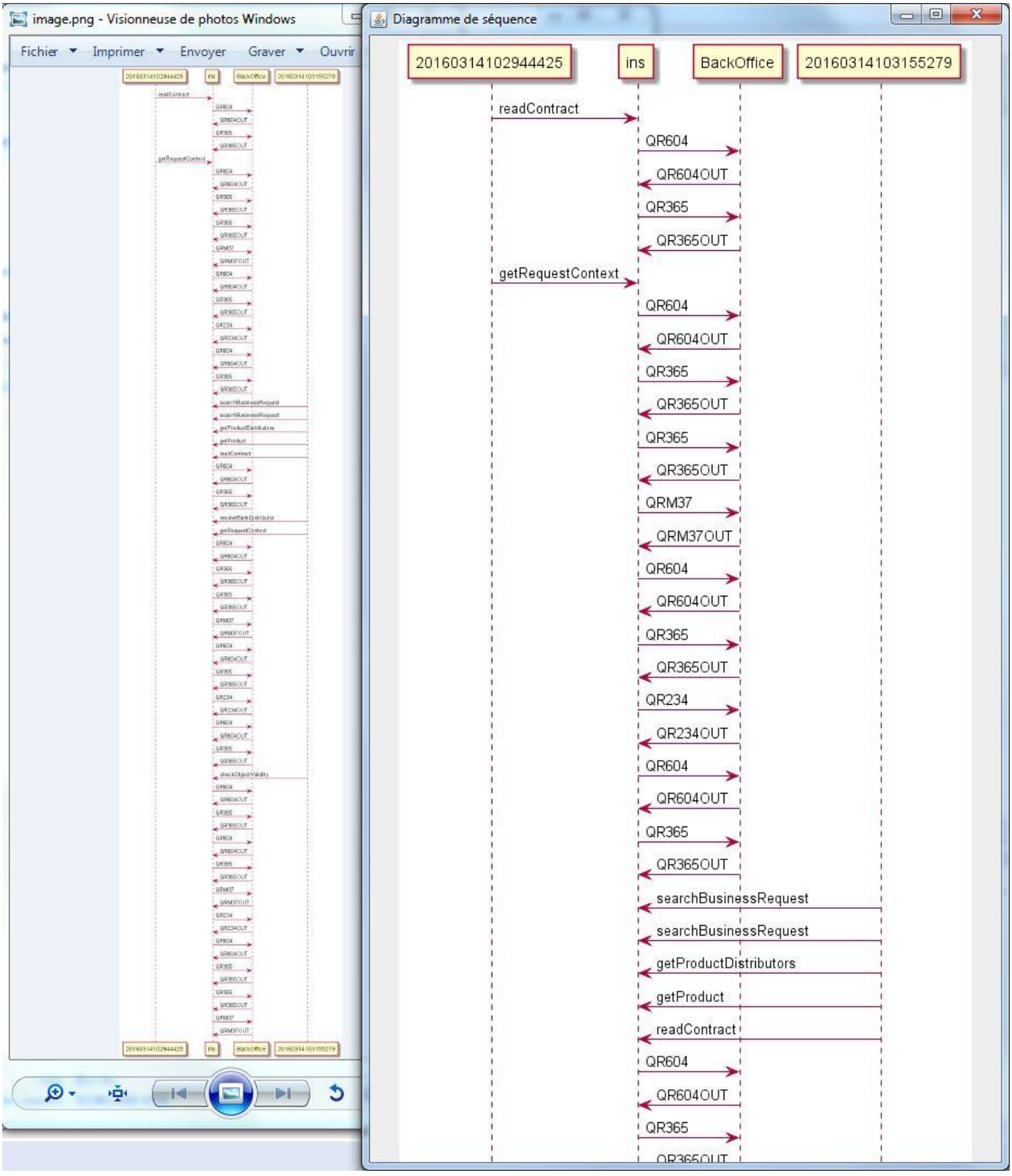
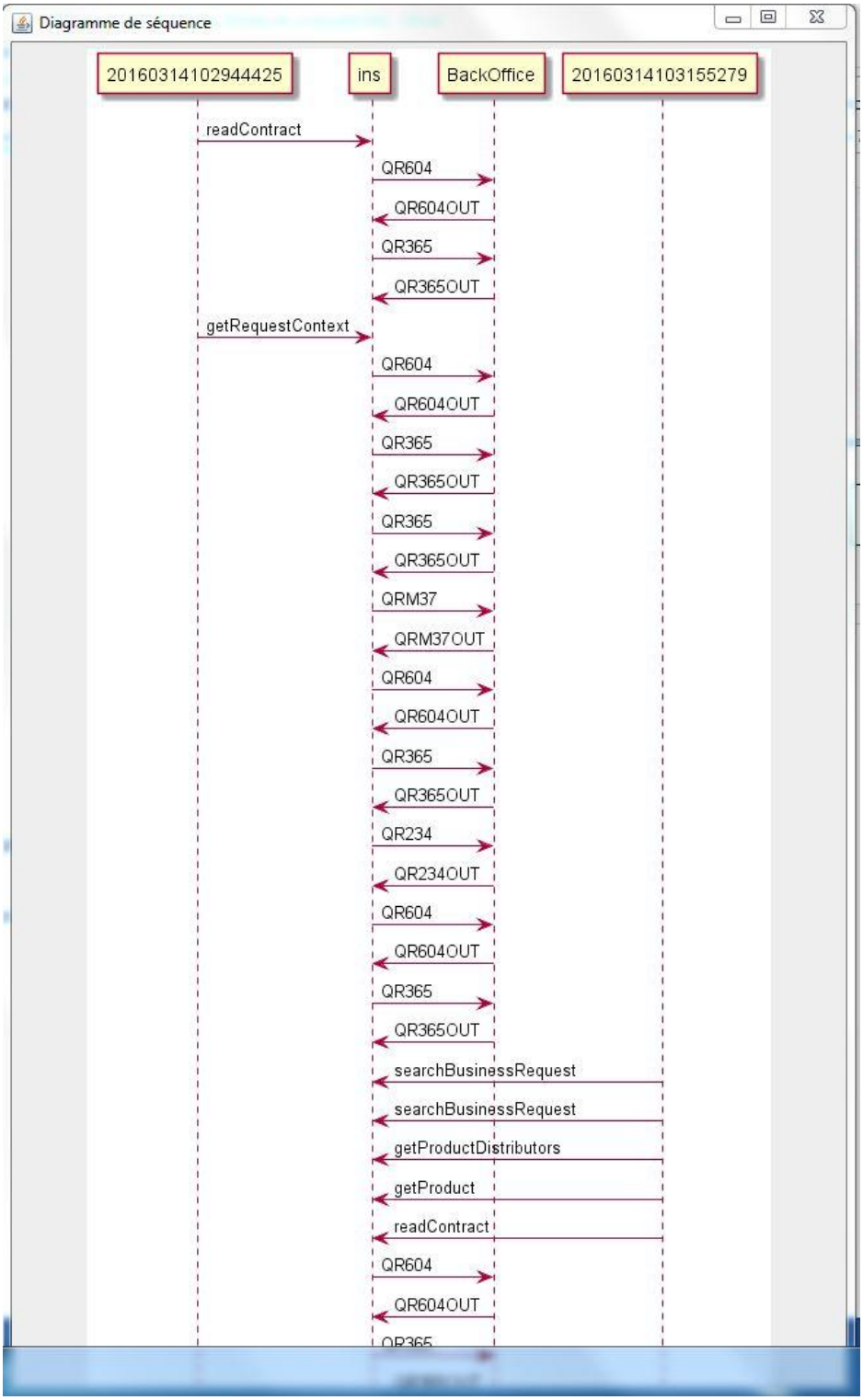


Diagramme généré par le programme et affiché dans une visionneuse d'image.

Ce même diagramme affiché par le programme.

b) *Problème 2*

Ce problème existait dans les captures d'écran du problème précédent mais n'était pas visible car la barre des tâches était cachée volontairement.
Ce problème est que la barre des tâches est affichée au-dessus du programme. Donc la partie basse de la fenêtre du programme est masquée par la barre de tâches. Voici ce que cela donne :

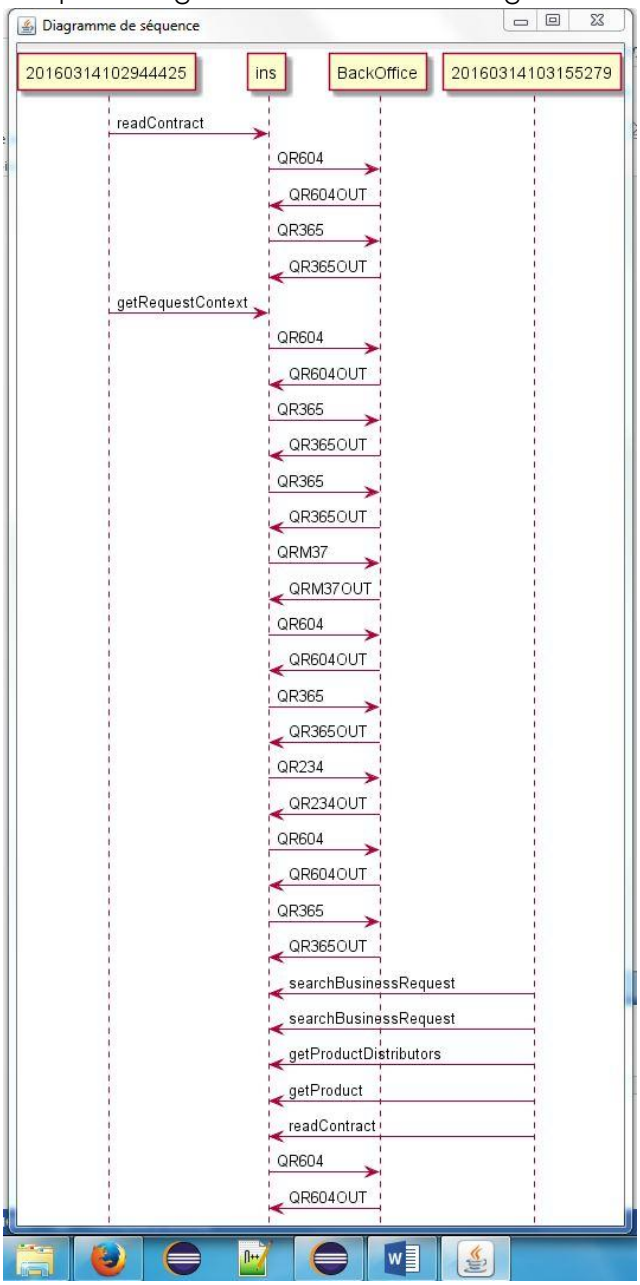


Il faut donc que la fenêtre du programme s'affiche par-dessus la barre de tâches de Windows.

Après quelques recherches, Java n'a pas l'air de pouvoir afficher les fenêtres au-dessus de la barre de tâches.

J'ai donc décidé de donner à la fenêtre du programme la hauteur utilisable de l'écran. C'est-à-dire la hauteur de l'écran moins la taille de la barre de tâches. Ainsi le bas de la fenêtre du programme s'arrêtera au niveau de la barre de tâches.

Par défaut j'avais réglé le programme pour que la fenêtre s'affiche en plein écran, avec la barre de tâche s'affichant par-dessus. Donc en plus de régler la hauteur de la fenêtre j'ai adapté la largeur de la fenêtre à la largeur de l'image affichée.

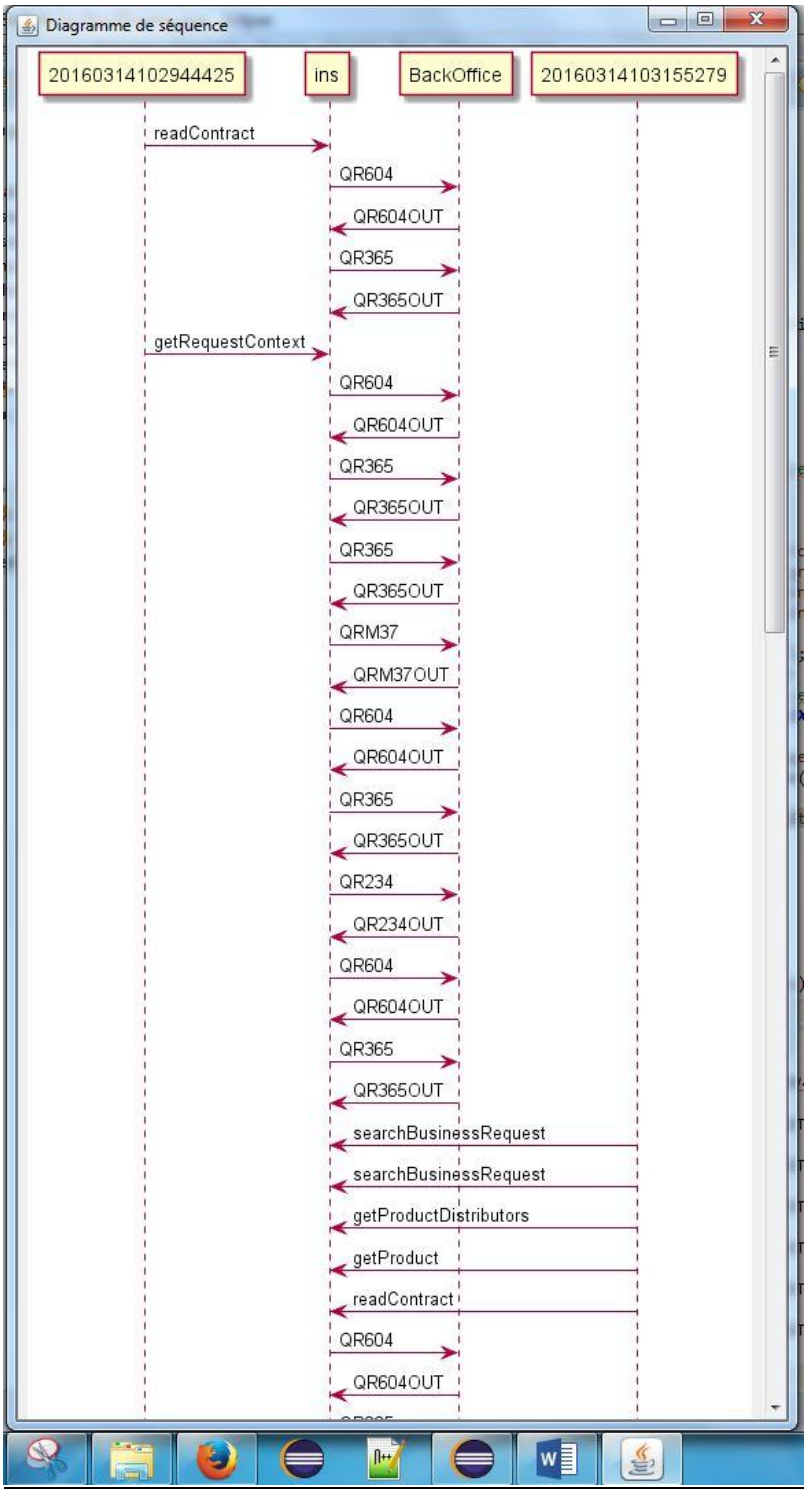


Résultat :

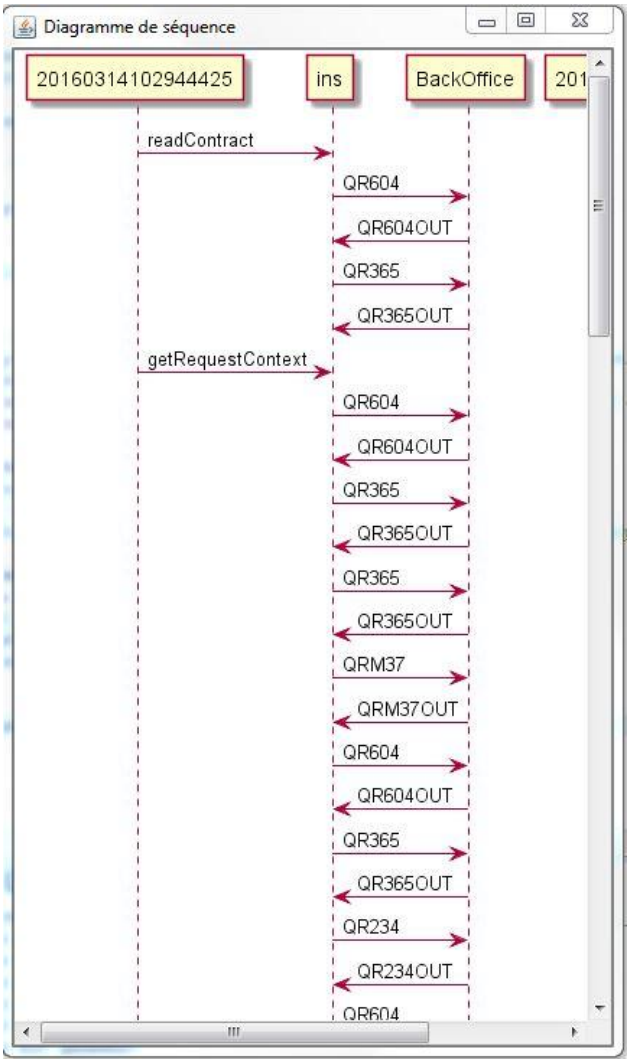
Le diagramme de séquence n'est plus enregistré sur le disque dur, il est transmis directement de la génération à l'affichage. Ainsi l'utilisateur n'a pas à chercher l'image sur son disque dur.

La fenêtre affichée tient compte de la largeur du diagramme et de la barre de tâches pour la hauteur.

Et des barres de défilement ont été ajoutées verticalement et horizontalement pour consulter la totalité du diagramme si ses dimensions sont trop grandes.



Barre horizontale :



c) *Problème n°3 :*

En testant sur un dossier de flux contenant récursivement plus de 700 échanges je me suis rendu compte qu'à peine 133 échanges étaient affichés dans le diagramme malgré que tous les échanges étaient détectés et affichés dans la console.

Après quelques tests sur d'autres dossiers de flux et recherches sur internet, j'ai trouvé sur la FAQ de PlantUML que la taille des images générées par la bibliothèque et limitée par défaut à un carré de 4096 pixels de côté.

Le problème se règle en changeant cette limite en lançant cette application dans la console Windows.
Ce problème n'avait pas été rencontré plus tôt car il n'y avait pas assez d'échanges dans les dossiers testés.

Liste des compétences :

A5.2.4 , Étude d'une technologie, d'un composant, d'un outil ou d'une méthode.

C5.2.4.1 Se documenter à propos d'une technologie, d'un composant, d'un outil ou d'une méthode.

➔ Lecture Javadoc et tutos sur l'utilisation de JFileChooser et les composants JFrame, JPanel et JScrollPane de la bibliothèque graphique Swing.

A5.2.3 Repérage des compléments de formation ou d'auto-formation utiles à l'acquisition de nouvelles compétences

C5.2.3.1 Identifier les besoins de formation pour mettre en oeuvre une technologie, un composant, un outil ou une méthode

C5.2.3.2 Repérer l'offre et les dispositifs de formation

➔ Lecture Javadoc et tutos sur l'utilisation de JFileChooser et les composants JFrame, JPanel et JScrollPane de la bibliothèque graphique Swing.

A4.1.7 Développement, utilisation ou adaptation de composants logiciels

C4.1.7.1 Développer les éléments d'une solution

C4.1.7.3 Analyser et modifier le code d'un composant logiciel

➔ Modification du programme.

d) *Problème n°4 :*

Le programme étant maintenant fonctionnel j'ai voulu le tester sur des plus gros dossiers de flux.
C'est à ce moment-là que je me suis rendu compte que le programme était de moins en moins rapide quand les dossiers étaient de plus en plus gros.
Le meilleur exemple est le dossier de flux contenant le plus de fichiers .log.

J'ai donc re-testé avec la version du programme qui n'affiche que la liste des fichiers en console.
Il fallait environ 15 minutes pour afficher les 784 fichiers .log présents récursivement sur 867 fichiers contre moins de 2 secondes avec une commande DOS que j'ai utilisée pour comparer car je trouvais ce temps beaucoup trop long.

J'ai cherché sur internet un code de parcours récursif de dossier pour voir si le problème venait de Java ou si mon code n'était pas optimisé.
J'ai testé le code trouvé sur le même dossier de flux et il lui a fallu moins de 2 secondes pour afficher toute la liste des fichiers présents récursivement.

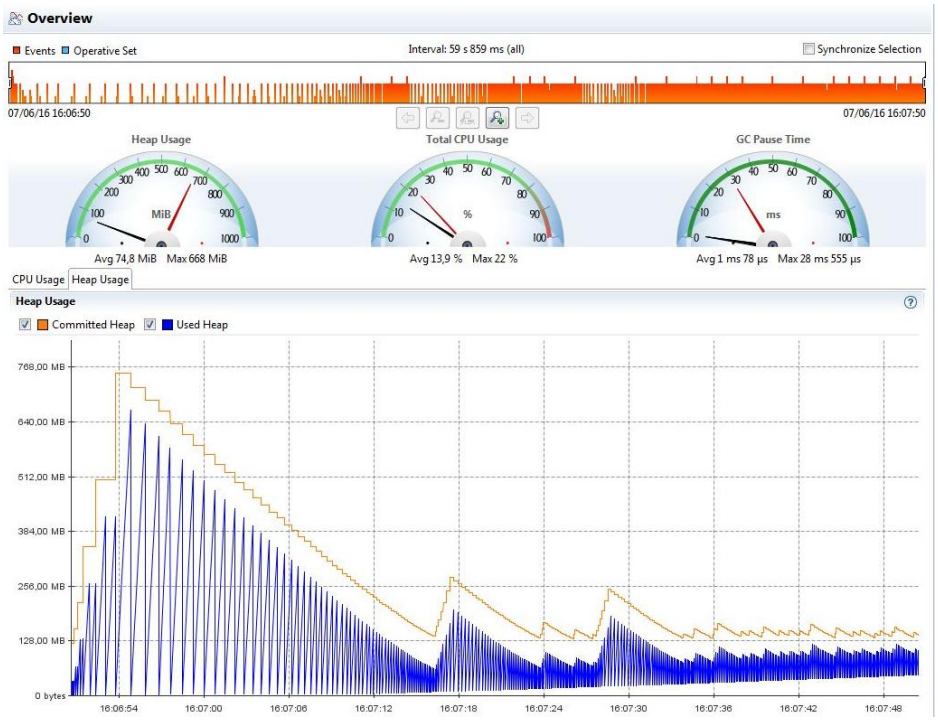
Le problème de rapidité venait donc de mon code.

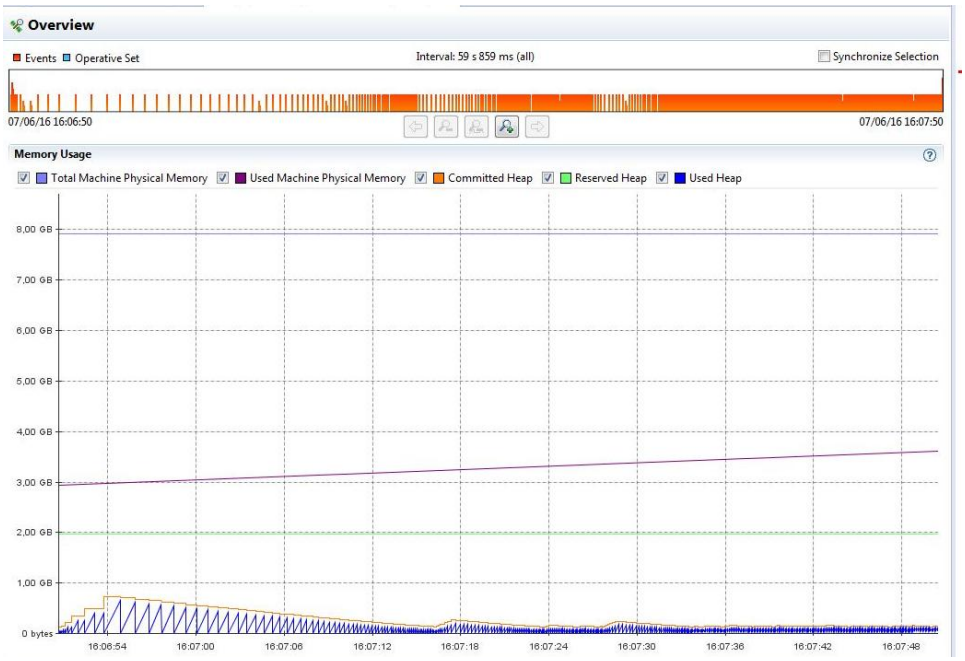
J'avais une classe parcourant les dossiers récursivement, créait des objets Application, Composant et ComposantExterne en fonction des dossiers ou fichiers rencontrés et renvoyait un tableau d'objets Application.

Mon programme principal parcourait récursivement les objets crées afin de les afficher à l'écran. Il y avait donc un parcours récursif dans un autre c'est-à-dire que pour chaque objet le programme parcourrait récursivement tous les dossiers.

J'ai utilisé le plugin Flight Recorder du logiciel Java Mission Control.
JMC permet la gestion, le suivi, le profilage et le dépannage d'applications Java.

Flight Recorder capture tout ce qui est exécuté dans l'environnement d'exécution Java par le programme à tester.

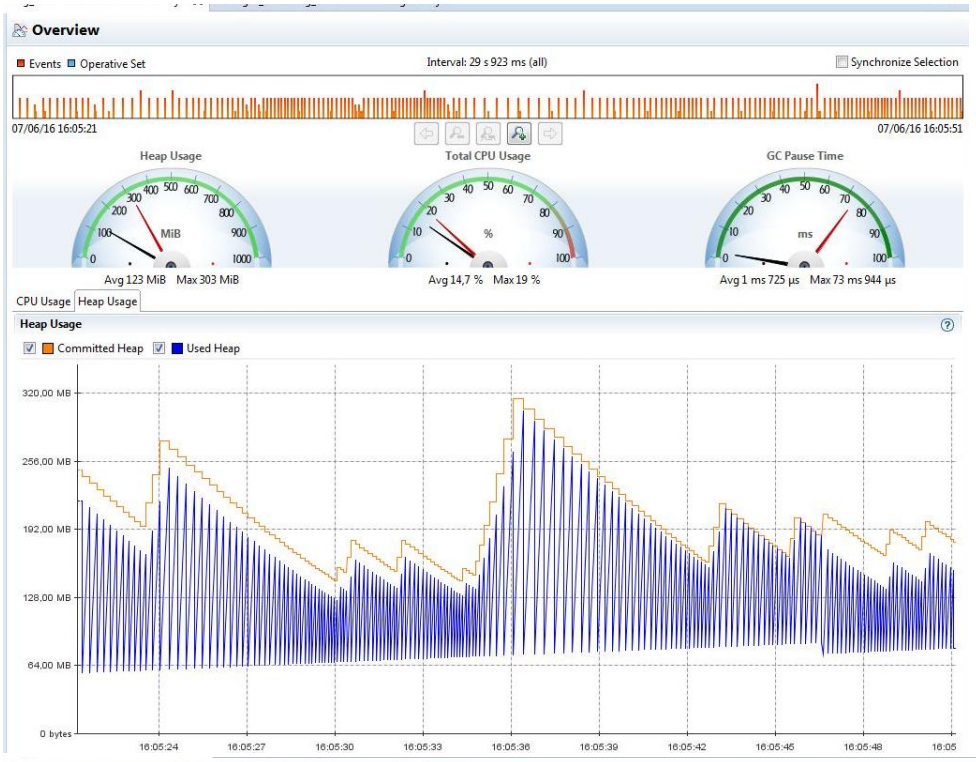




Sur le premier graphique on peut voir que l'utilisation du heap est assez importante (668 Mib) lors du plus gros pic et que la plus forte utilisation du processeur est de 22%.
Tous les objets créés sont obligatoirement stockés dans le tas (heap).

Sur le deuxième graphique la mémoire physique utilisée par la machine est en constante augmentation, d'un peu moins de 3GB à environ 3,6GB utilisé et ce n'était que le début de l'exécution du programme.

La solution pour éviter la lenteur du programme a été de fusionner la classe qui parcourait les dossiers et le programme principal afin de n'avoir qu'un seul parcours récursif.
Le programme étant trop rapide pour être détecté par JMC, je l'ai placé dans une boucle infinie pour les besoins du test.



Sur le premier graphique on peut voir que l'utilisation du heap est moins importante (303 Mib) lors du plus gros pic et que la plus forte utilisation du processeur est de 19%.

Sur le deuxième graphique la mémoire physique utilisée par la machine reste à environ 3,2GB

Donc après modification, le programme utilise moins de ressources et est beaucoup plus rapide puisqu'il met moins de 2 secondes pour faire ce que faisait l'ancienne version en plus de 15 minutes.

Liste des compétences :

A2.3.1 Identification, qualification et évaluation d'un problème
- C2.3.1.1 Repérer une suite de dysfonctionnements récurrents d'un service
- C2.3.1.2 Identifier les causes de ce dysfonctionnement
- ➔ Modification du programme

A4.2.1 , Analyse et correction d'un dysfonctionnement, d'un problème de qualité de service ou de sécurité
- C4.2.1.2 Repérer les composants à l'origine du dysfonctionnement
- C4.2.1.3 Concevoir les mises à jour à effectuer
- C4.2.1.4 Réaliser les mises à jour
- ➔ Modification du programme.

3.5 INTEGRATION AVEC LES OUTILS EXISTANTS

3.5.1.1. MAVEN

Maven est un outil de construction de projets (build) open source développé par la fondation Apache. Il permet de faciliter et d'automatiser certaines tâches de la gestion d'un projet Java.

Il permet notamment :

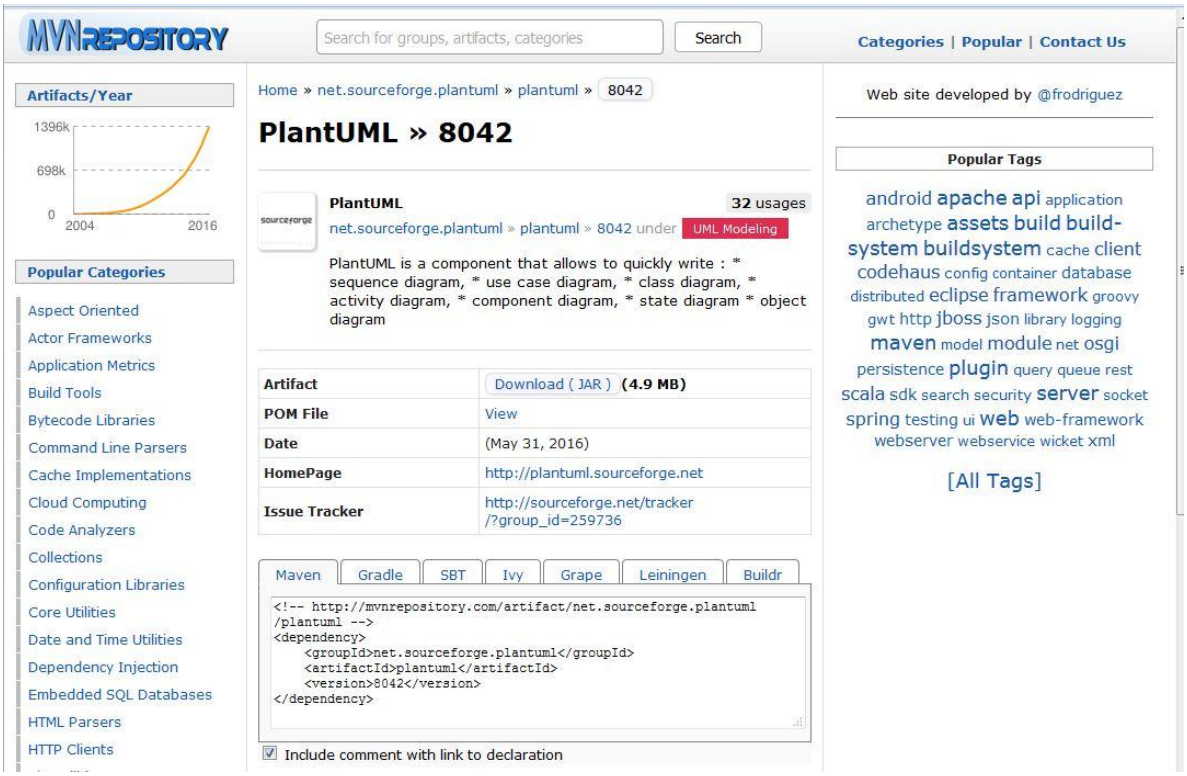
- d'automatiser certaines tâches : construction, compilation, tests unitaires, mise à jour et déploiement des applications qui composent le projet
- de gérer des dépendances vis-à-vis des bibliothèques nécessaires au projet
- de générer des documentations concernant le projet

Un plugin Maven est installé par défaut dans Eclipse mais je l'ai aussi l'installé sur l'ordinateur.

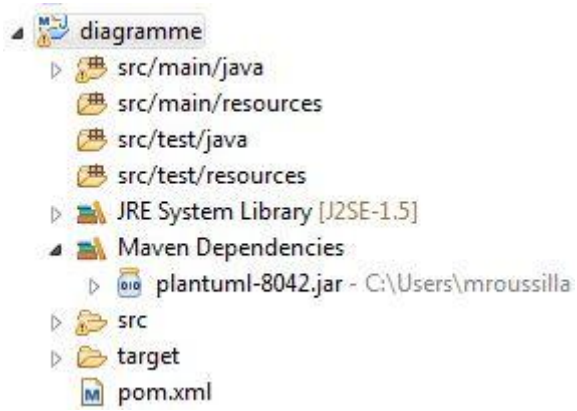
Le programme auquel ma solution sera reliée utilise Maven. Je dois donc maveniser mon programme afin de pouvoir l'intégrer.

La description d'un projet est faite dans un fichier XML nommé POM (Project Object Model). Cette description contient notamment les dépendances, les spécificités de construction (compilation et packaging), éventuellement le déploiement, la génération de la documentation, l'exécution d'outils d'analyse statique du code, ...

J'y ai donc renseigné la dépendance à la bibliothèque PlantUML comme indiqué dans le Maven Repository.



Une fois la dépendance renseignée, Maven a automatiquement téléchargé la version demandée de la bibliothèque PlantUML. Il télécharge aussi les dépendances dont PlantUML a besoin. On peut retrouver ces dépendances dans Maven Dependencies.



Liste des compétences :

A4.1.6 Gestion d'environnements de développement et de test

C4.1.6.1 Mettre en place et exploiter un environnement de développement

➔ Installation de Maven.

A5.2.4 , Étude d'une technologie, d'un composant, d'un outil ou d'une méthode.

C5.2.4.1 Se documenter à propos d'une technologie, d'un composant, d'un outil ou d'une méthode.

➔ Recherches sur les agents Java.

3.5.1.2. LES AGENTS JAVA

On appelle agent une entité physique ou virtuelle qui possède tout ou partie des fonctionnalités suivantes :
- est capable d'agir dans un environnement
- peut communiquer avec d'autres agents
- est mue par un ensemble de tendances
- possède des ressources propres
- est capable de percevoir son environnement
- possède des compétences et offre des services
- dont le comportement tend à satisfaire ses objectifs.

Les agents permettent d'instrumenter des programmes s'exécutant dans une JVM (Java Virtual Machine qui est un appareil informatique fictif qui exécute des programmes compilés sous forme de bytecode Java) en interceptant le chargement des classes et en modifiant directement le bytecode si nécessaire.

Un agent est chargé lors du démarrage de la JVM, accompagné ou non d'options. La JVM est belle et bien consciente de son existence, si le chargement d'un agent échoue, l'exécution du programme est interrompue.

Le but est de créer un agent Java permettant de relier ma solution au programme de tests existant.
Ainsi lorsque le programme de tests aura fini les tests, mon programme sera lancé et affichera automatiquement le diagramme de séquences correspondant. Le choix de dossier par une fenêtre ne servirait plus dans mon programme.

Il reste à réaliser l'intégration avec le programme de tests existant. Je n'ai pas eu accès au programme utilisé par le client car il m'aurait fallu une machine virtuelle CNP, il m'était donc impossible de réaliser cette partie.

J'ai réalisé une documentation utilisateur et commenté le code du programme pour les personnes qui intègreront mon programme aux composants utilisés par les clients.

Liste des compétences :

A4.1.10 Rédaction d'une documentation d'utilisation
C4.1.10.1 Rédiger la documentation d'utilisation, une aide en ligne, une FAQ
➔ Rédaction de la documentation utilisateur.