Tuteur professionnel : MAUNY Éric Date du stage : du 07 avril au 22 juin 2023

Tuteur pédagogique : HADJ AMOR Khaoula

Rapport de stage

Amélioration de l'outil SSAM : Ajout d'un support d'import/export de fichiers Excel

ALPHARE-FASIS **24 Avenue Georges Brassens** **Bâtiment A, 1er étage** **31700 Blagnac**	**Signature du tuteur :** 

Remerciements

Je tiens tout d'abord à remercier Monsieur Éric MAUNY, mon responsable de stage, ainsi que Christine CHANSSARD, pour leur accueil, leur patience et leur implication dans mon travail tout au long de mon stage.

Pour finir je voudrais remercier toutes les personnes qui m'ont accueilli et intégré à l'entreprise, pour leur gentillesse et leur bonne humeur.

Sommaire :

Table des matières

Glossaire :

- Sheet : feuille de calcul d'un fichier Excel
- MVC : Modèle Vue Contrôleur. Architecture très utilisée dans la programmation d'un logiciel utilisant les interfaces graphiques. Elle a l'avantage d'être claire et facilement maintenable pour les programmeurs.
- SSAM : Schématisation des Scénarios d'Accident Majeur. Outil interne à l'entreprise pour réaliser des graphes permettant de synthétiser l'analyse de risques d'une installation (également appelés nœuds papillon).
- Swing : bibliothèque de Java permettant de créer des interfaces graphiques.
- Workbook : Collection contenant une ou plusieurs feuilles de calcul Excel
- HSSF : Horrible Spreadsheet Format
- XSSF : XML Spreadsheet Format ou Format de feuilles de calcul XML

Introduction :

Dans le cadre de ma 2ème année de BUT Informatique, j'ai dû effectuer un stage de 11 semaines. Pour moi ce stage n'était pas un premier contact avec le monde professionnel ayant déjà effectué un stage pour ma 2ème année de DUT Informatique.

Le stage s'est déroulé au sein de l'entreprise ALPHARE-FASIS, du 7 avril au 22 juin 2023. L'objectif du stage avait pour objectif d'ajouter une fonctionnalité d'import d'un fichier Excel dans leur application SSAM. Cet objectif ayant été réalisé plus rapidement que prévu, d'autres tâches détaillées plus bas m'ont été confiées. Ce stage a été effectué en autonomie principalement car l'entreprise n'a pas de service informatique. Cette application est développée en Java.

Ce document a pour but de présenter de façon détaillée le déroulement du stage. Il est organisé comme suit :

- Présentation succincte de l'entreprise qui m'a accueilli
- Présentation du sujet de stage et de la tâche planifiée au début mais aussi des tâches qui m'ont été affectées après la fin de la tâche principale
- Présentation du travail effectué
- Bilan sur les compétences acquises au cours de ce stage

I. Présentation Entreprise

La société ALPHARE-FASIS, née en 2013 par la fusion de deux sociétés, ALPHARE et FASIS, est un bureau d'études et de conseils dans les domaines de la sécurité et de l'environnement industriels.

Basée à Blagnac (Haute-Garonne), ALPHARE-FASIS déploie son activité sur l'ensemble du territoire national, métropolitain, d'outre-mer ainsi qu'à l'étranger.

ALPHARE-FASIS met ses compétences et son savoir-faire au service d'industriels, d'ingénieries, de bureaux d'études et de collectivités locales. Ses profils variés permettent de garantir sa capacité à réaliser des prestations sur divers sujets dans les domaines de la sécurité et de l'environnement.

Afin d'améliorer la qualité de son travail et sa performance, ALPHARE-FASIS a besoin de développer/optimiser en interne des outils performants, notamment sur la thématique des études de dangers. L'outil SSAM en fait parti.

ALPHARE-FASIS est une société par action simplifiée (SAS) au capital de 401 646 € dont l'actionnariat est indépendant. L'actionnaire majoritaire est Christine CHANSSARD, présidente et fondatrice d'ALPHARE-FASIS.

1. Les métiers

Les principaux domaines d'activités d'ALPHARE-FASIS sont la réalisation d'études (de dangers, d'impact…) et de prestation de conseils. Les activités peuvent être divisées en 3 catégories :

- ETUDES
  - Études de dangers,
  - Études d'impacts,
  - Dossiers réglementaires en lien avec les Installations Classées pour la Protection de l'Environnement (ICPE)
  - Analyse des risques professionnels,
  - Etc.

- CONSEILS
  - Management,
  - Audit, Formation,
  - Veille réglementaire personnalisée,
  - Transport des matières dangereuses (Conseil à la sécurité).

- INSPECTIONS
  - Contrôle périodique de certaines catégories d'installations classées soumises à déclaration.

2. Outils utilisés

Pour réaliser les études de dangers et les analyses de risques avec plus d'efficacité et de qualité, l'équipe utilise des outils internes à l'entreprise. On distingue deux catégories : les outils de modélisation et les outils de schématisation.

Outil de modélisation

Pour l'évaluation des conséquences sur l'environnement des scénarios d'accidents dans le cadre des études de dangers :

- PHAST
- EFFECTS
- Winvent
- Etc. 6

Pour les scénarios d'exposition environnementale dans le cadre de l'évaluation des risques sanitaires des études d'impacts :

- ADMS
- HESP
- Screen
- Etc.

Outil de schématisation des scénarios d'accidents majeurs (SSAM 1.5)

Les phénomènes accidentels peuvent être représentés sous forme de nœud papillon. Ce formalisme permet de visualiser l'ensemble des scénarios pouvant mener à ce phénomène et de représenter les barrières permettant de l'éviter ou d'en réduire les conséquences.

Cette représentation peut avoir plusieurs intérêts :

- La visualisation rapide et synthétique du scénario,
- La justification des barrières de sécurité, nommées Mesures de Maîtrise des Risques,
- La prise en compte des conjonctions d'événements simples.

3. Les clients

La société travaille majoritairement pour des sites industriels. Les principaux clients sont :

- TotalEnergies,
- Arianegroup,
- AIRBUS France,
- AIR France,
- FINAGAZ,
- SFR,
- Etc.

II. Présentation du sujet de stage

1. Contexte de l'utilisation de l'outil SSAM

Logo de SSAM

L'outil SSAM est un outil interne développé en Java par de précédents stagiaires, la première version ayant été réalisée en 2003 et la dernière modification en 2016.

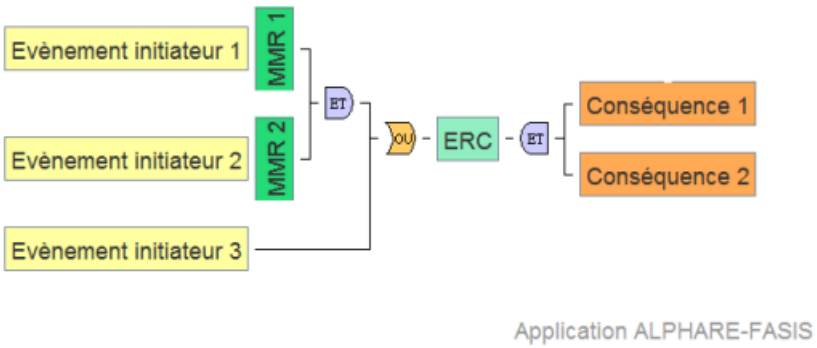
Cet outil est utilisé par l'équipe d'ALPHARE-FASIS pour réaliser des nœuds papillons associés à des phénomènes dangereux, majoritairement dans le cadre de la réalisation d'une étude de danger pour un site industriel. Cette étude peut être réalisée aussi bien sur une installation existante que sur un projet.

L'importation de nœuds papillons se faisait uniquement via fichier XML exporté au préalable, De même, l'exportation des nœuds papillons se faisait uniquement via une image png ou un fichier XML.

Les accidents majeurs sont généralement la conséquence d'un enchaînement d'événements indésirables combinés à des défaillances de mesure de maîtrise des risques. Une analyse de risque permet de mettre en évidence un événement central (événement sur lequel on réalise l'étude), les événements initiateurs (cause), les conséquences et les mesures de maîtrise des risques qui sont mises en place afin de diminuer la probabilité d'occurrence du phénomène accidentel.

L'outil SSAM permet de schématiser de manière détaillée le déroulement chronologique d'un accident sous forme d'une arborescence en nœud papillon. Au centre se trouve l'ERC (Événement Redouté Central), à gauche les événements initiateurs et à droite les conséquences possibles issues de l'ERC. Cette schématisation permet aussi de visualiser les mesures de maîtrise des risques (MMR) appliquées sur chaque événement.

Les événements sont reliés entre eux par des portes logiques « ET » et « OU » et chaque composant du schéma peut se voir attribuer une couleur, un descriptif plus détaillé ainsi qu'une probabilité.



Exemple de nœud papillon

2. Objectif du stage

L'objectif du stage était de mettre en place un système d'import de fichier Excel dans un premier temps.

Cette tâche ayant été terminée en avance, d'autres tâches m'ont été confiées par la suite.

I. Import de fichier Excel

L'objectif de base du stage était d'implémenter une nouvelle fonctionnalité qui permet d'importer un fichier Excel pour créer des nœuds papillons.

II. Export de fichier Excel

Le second objectif confié (une fois le premier terminé) est d'implémenter une fonctionnalité complémentaire à la première qui est l'export d'un fichier Excel à partir d'un nœud papillon déjà créé.

III. Travail réalisé

1. Outils utilisés

L'outil SSAM a été développé dans le langage de programmation java. Ce langage a été repris pour les modifications effectuées.

Afin de développer avec plus d'aisance et d'efficacité, j'ai utilisé l'environnement de développement Eclipse pour programmer en Java.

2. Analyse de l'existant

Avant de commencer toute modification du logiciel, une étape d'analyse de l'existant a été réalisée. Au démarrage du stage, plusieurs ressources ont été mises à disposition :

- La version 2.0 de SSAM (dernière version utilisée par l'équipe d'ALPHARE-FASIS),
- Le code source de SSAM 2.0,
- Le manuel utilisateur,
- Un fichier texte contenant des conseils donnés par le précédent stagiaire sur la façon de modifier SSAM.

I. Première utilisation du logiciel

Une première utilisation du logiciel a permis de comprendre son fonctionnement, de prendre connaissance de l'interface (voir Annexe 1) et des fonctionnalités proposées et de se faire une première idée sur la façon dont il a été conçu.

II. Prise de connaissance des notes du précédent stagiaire

Les conseils donnés par le précédent stagiaire donnaient des indications sur la façon dont était structuré le logiciel, ainsi que sur les fichiers obligatoires à avoir lors de l'installation de SSAM 2.0 sur un poste. En effet, pour que le logiciel fonctionne, il avait besoin de deux fichiers :

- Un fichier properties : ce fichier comportait des paramètres utilisés par SSAM, comme le titre du logiciel ou encore le chemin vers le répertoire de travail par défaut ;

- Un fichier liste : ce fichier contenait un ensemble de noms définissant les événements initiateurs, les conséquences et les mesures de maîtrise de risques. Il agissait comme un dictionnaire que l'utilisateur utilisait au sein de l'outil pour nommer ces éléments.

III. Étude du code source

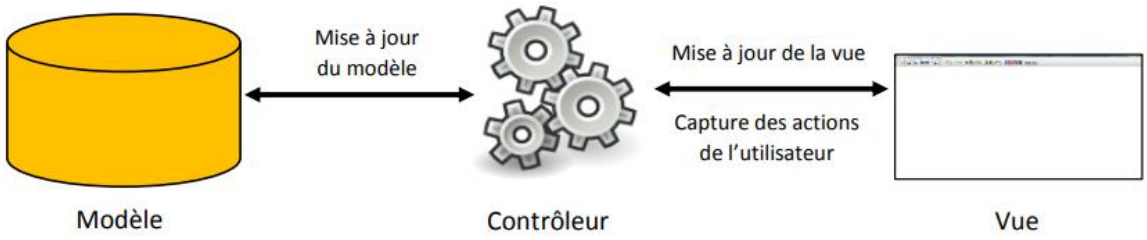
À la suite de la prise de connaissances de toutes ces informations, l'étude du code source a été établie. Le projet était assez conséquent : il contenait environ 160 fichiers et approchait les 16 000 lignes de code. L'identification du rôle de chaque fichier était importante afin de comprendre l'architecture du logiciel et de pouvoir réaliser les modifications avec plus de simplicité tout en respectant l'architecture utilisée. Il est possible de classer ces fichiers en 3 catégories :

- La vue : ensemble des fichiers servant à créer l'interface graphique. C'étaient ces fichiers qui allaient créer la fenêtre principale, le menu, les images, etc... c'est-à-dire ce qui est visible pour l'utilisateur ;

- Le modèle : ensemble des fichiers servant à représenter les données de manière interne, invisible à l'utilisateur. Ces fichiers étaient tous aussi importants car ils définissaient les informations qui seraient par la suite affichées pour l'utilisateur. On note que lorsque l'on sauvegardait un nœud papillon, les données sauvegardées étaient structurées selon ces fichiers ;

- Le contrôleur : ensemble des fichiers qui allait faire le lien entre la vue et le modèle. Par exemple, lorsque l'utilisateur réalisait une action (sur la vue), le contrôleur capturait l'action et modifiait le modèle en conséquence. On remarque que le projet suivait ce que l'on appelle le modèle MVC (Modèle, Vue, Contrôleur), qui est un modèle souvent utilisé lors de réalisation d'applications utilisant les interfaces homme-machine.

Le schéma ci-dessous représente les liens entre les différentes.



Entités :

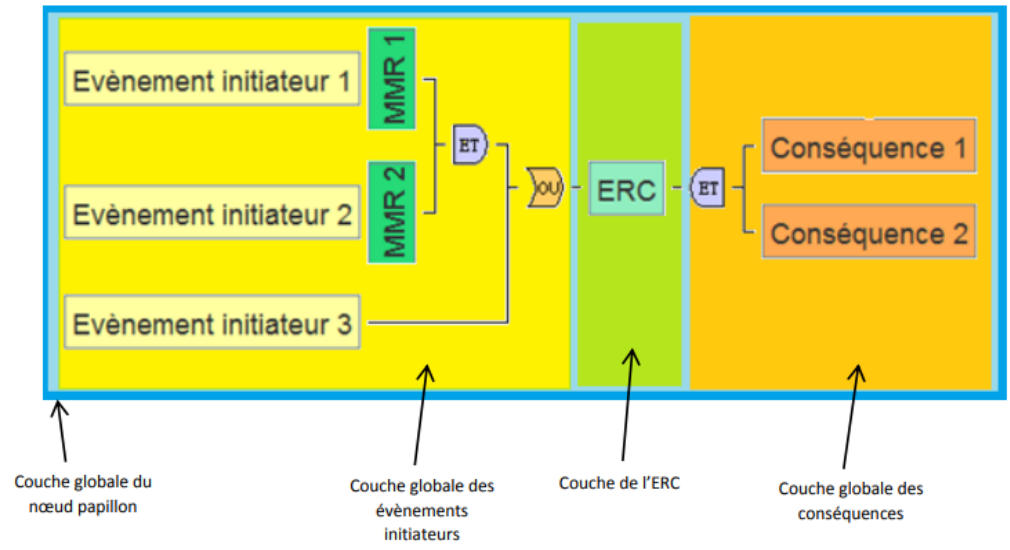
Ce modèle avait pour avantage d'être clair sur l'architecture qu'il imposait, d'être plus souple et de simplifier la mise en place de modifications du code. Par exemple, si l'on souhaitait modifier complètement l'affichage, on pouvait créer une nouvelle vue (avec un nouveau contrôleur adapté) sans toucher au modèle. Cela permettait aussi de revoir une vue existante sans avoir à se préoccuper de l'impact que cela aurait sur le modèle.

Tout ce qui se rapportait à la vue (la fenêtre principale, le menu et même le nœud papillon) se basait sur la bibliothèque Swing de Java. Cette bibliothèque contenait des composants nécessaires à la construction d'une interface graphique. Chaque élément du nœud papillon était alors construit par des boutons, avec un fond coloré selon la nature de l'élément (événements, mesures, portes logiques...) et un libellé permettant de l'identifier.

Pour mieux comprendre comment se construisait le nœud papillon, on raisonne par l'utilisation d'un système de couches, chaque couche permettant de positionner les composants à sa manière. Modèle Contrôleur Vue Mise à jour de la vue Capture des actions de l'utilisateur Mise à jour du modèle 16.

Niveau global

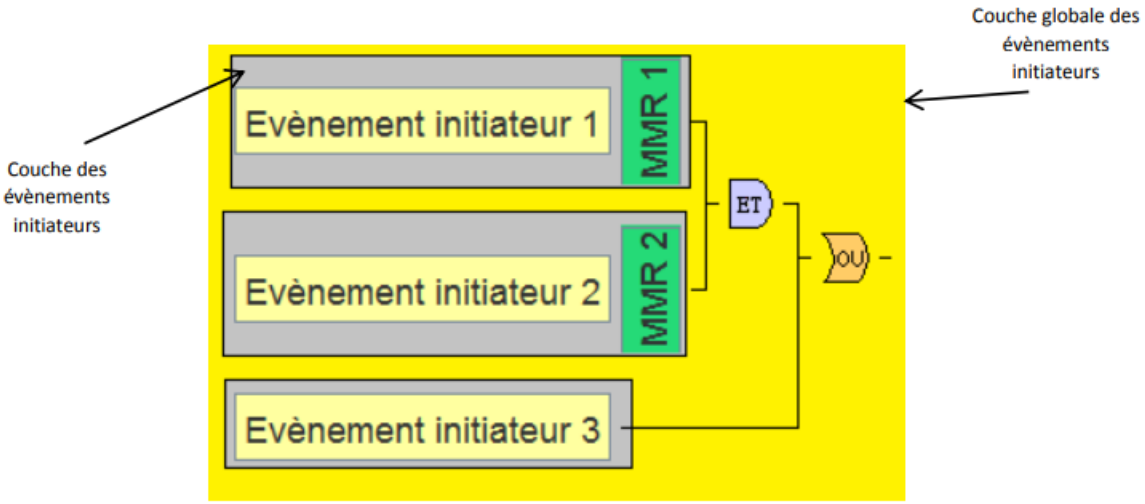
Le schéma ci-dessous représente l'agencement des éléments du nœud papillon à un niveau global :



- Les événements initiateurs et les barrières associées sont contenus dans la couche jaune ;
- L'ERC et les barrières associées sont contenus dans la couche verte ;
- Les conséquences et les barrières associées sont contenues dans la couche orange ;
- L'encadré bleu est la couche englobante. Il contient les trois couches précédentes.

Niveau des couches globales des événements

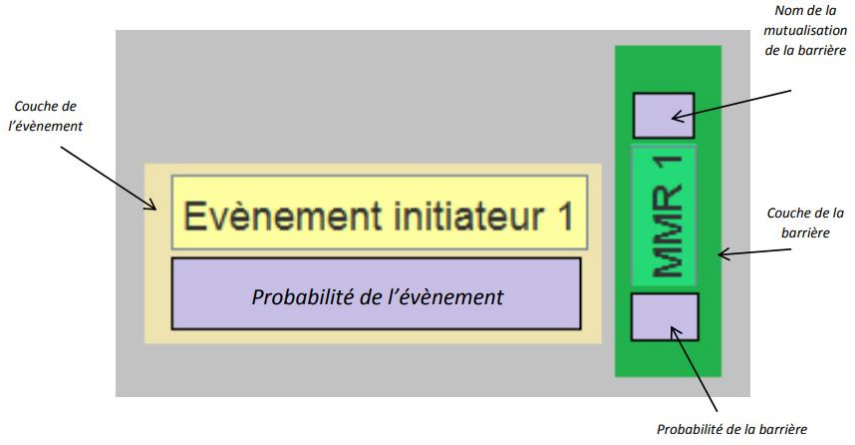
Le schéma ci-dessous représente l'agencement des événements au sein de la couche globale des événements initiateurs. Ce schéma s'applique aussi sur la couche globale des conséquences :



- Chaque événement initiateur et ses barrières associées sont contenus dans la couche grise
- Les portes logiques sont directement placées sur la couche jaune

Niveau des couches d'un événement

Le schéma ci-dessous représente l'agencement des barrières autour d'un événement. Ce schéma s'applique aussi aux conséquences et à l'ERC :



- Un événement et éventuellement sa probabilité (encadré violet) sont contenus dans la couche jaune.
- Une barrière et éventuellement sa probabilité et le nom de sa mutualisation (encadré violet) sont contenus dans la couche verte.

3. Résultats obtenus

i. Import d'un fichier Excel

Comme vu dans la partie II.1, l'importation/création d'un nœud papillon se faisait soit à la main, soit via un fichier XML exporté au préalable. Mon travail a consisté à rajouter une option d'importation facile via des feuilles de calcul Excel, ce qui permettrait à l'entreprise de créer les nœuds papillons au préalable, puis de les importer directement dans l'application.

Chemin d'importation d'un nœud papillon avant la mise à jour :

Chemin d'importation d'un nœud papillon après la mise à jour :

Dès lors, mon travail a consisté à rajouter une option supplémentaire pour l'importation d'un nœud papillon dans l'application.

1. Choix de la technologie à utiliser

Pour ce qui est de la technologie utilisée pour lire et écrire dans des fichiers Excel, j'ai pris la décision d'utiliser Apache POI qui est une bibliothèque gratuite qui permet de lire et écrire dans des fichiers Excel mais aussi d'autres types de fichiers de Microsoft comme des PowerPoint, Word…

J'ai fait le choix d'utiliser Apache POI pour une raison simple : ma connaissance de la bibliothèque et mon expérience déjà acquise dans son utilisation.

2. Utilisation de la bibliothèque Apache POI

Pour tous les fichiers Excel, il existe 2 types de classes différents :

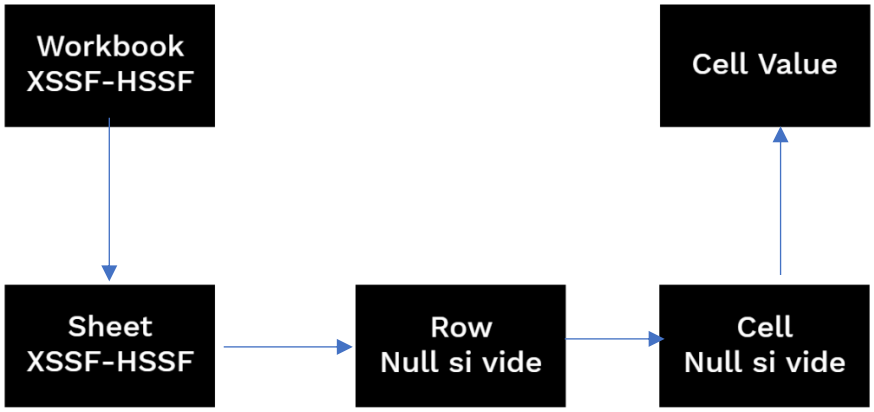
- HSSF : (pour les fichiers Excel possédant une extension de type xls qui correspond au fichier Excel pré 2009) ;
- XSSF : (pour les fichiers Excel possédant une extension de type xlsx qui correspond au fichier Excel post 2009).

Cette bibliothèque fonctionne en récupérant le workbook (qui est notre fichier Excel) depuis un FileInputStream de java qui contient le lien de notre fichier.

La création d'un workbook peut nous sortir une exception qui est une exception de type IOException. Cette exception est levée quand il y a un problème avec le FileInputStream (mauvaise lecture du Stream par exemple) ou que le fichier n'est pas un fichier Excel.

Par la suite, le workbook précédemment récupéré contient des Sheets qui correspondent à nos feuilles de calcul.

Représentation du fonctionnement d'un fichier Excel dans Apache POI :

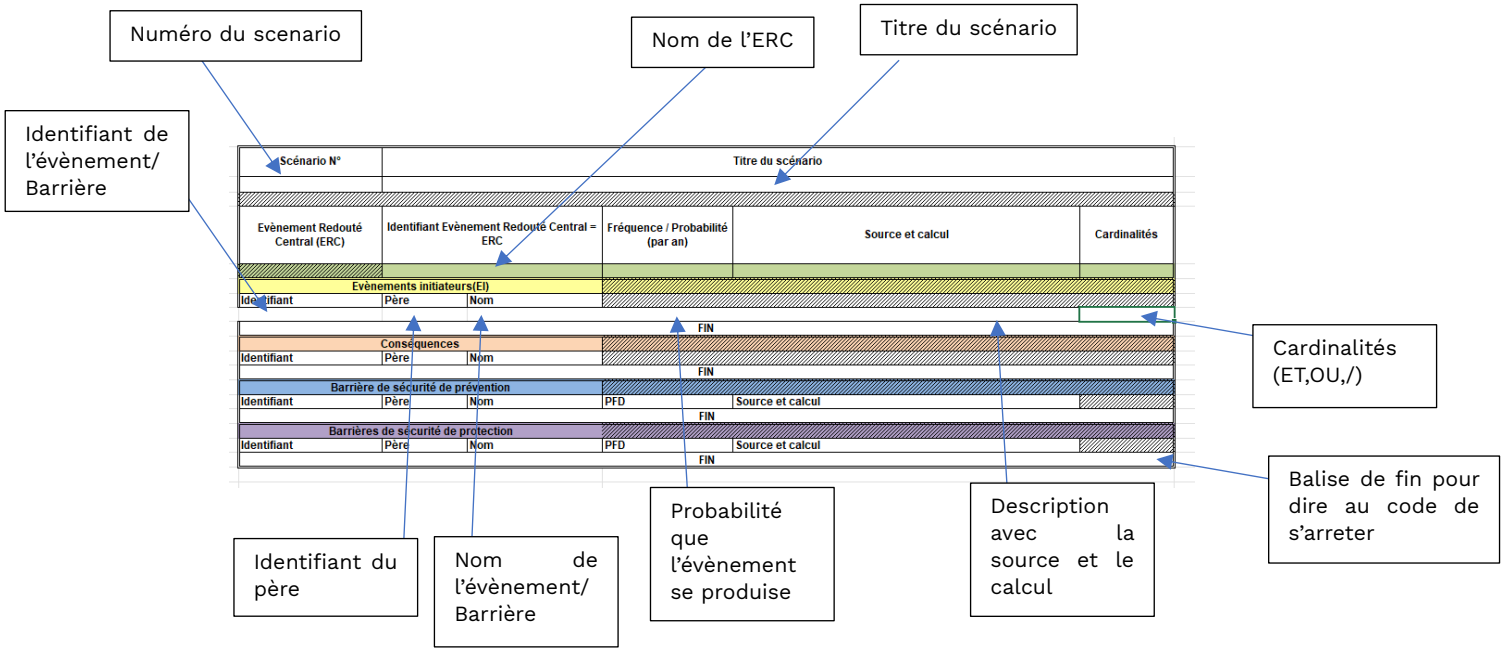


Pour pouvoir récupérer les données qui sont contenues par notre fichier Excel nous devons dans un premier temps récupérer le workbook via l'utilisation des FileInputStream de Java, Ensuite dans un second temps via le nom ou un index nous devons récupérer la Sheet pour pouvoir ensuite itérer à l'intérieur afin de récupérer les données voulues.

3. Mise en place d'un format de fichiers Excel

Après avoir choisi la bibliothèque, j'ai mis en place un format de fichier qui me faciliterait le code à produire derrière tout en étant simple pour que le fichier ne soit pas trop compliqué à remplir par les utilisateurs.

Voici le format de fichier après validation auprès de mon tuteur :



Grâce à ce format, je peux facilement savoir où se situe chacun des événements (bien sûr il peut y avoir des erreurs).

Par exemple :

Scénario N°	scénario				
3	BLEVE camions				
Evènement Redouté Central (ERC)	**Identifiant Evènement**		**Fréquence / Probabilité (par an)**	**Source et calcul**	**Cardinalités**
	BLEVE camions				
Evènements initiateurs(EI)					
Identifiant	**Père**	**Nom**			
EI1	ERC	Test	1.00E+04		/
EI2	ERC	Test2	1.00E-05		*ou*
EI3	EI1	Test3	1.00E-04		/
EI4	EI1	Test4	1.00E-04		*et*
EI5	EI2	Test5	1.00E-04		/
EI6	EI2	Test6	1.00E-04		*ou*
EI7	EI5	Test7	1.00E-04		/
EI8	EI6	Test8	1.00E-04		/
EI9	EI10	Test9	1.00E-04		/
EI10	EI8	Test10	1.00E-04		/
		FIN			

Lors de la lecture de ce fichier, une erreur sera levée et un message d'erreur sera retourné à l'utilisateur car le père EI10 est pour l'instant inconnu dans notre système donc l'exception spécial PereInconnu sera levée.

Une fois la mise en place du format de fichiers Excel voulu je me suis attaqué au développement de la fonctionnalité d'import.

4. Développement de la fonctionnalité d'import

Après avoir fini d'étudier le code qui m'a été donné, de choisir la technologie que j'allais utiliser pour lire des fichiers Excel, et de mettre en place un format de fichier Excel, je commence enfin le développement de la fonctionnalité. Dans un premier temps, j'ai créé le fichier ActionImporterExcel qui correspondra à l'action qui allait se passer lors du clic sur le bouton, en laissant le fichier vide.

Ensuite, je suis passé à la création du bouton en lui-même qui a été assez simple, car plusieurs autres boutons étaient déjà présents.

Je crée mon action en lui passant en paramètre l'icône voulu.

Ensuite, je crée mon item qui sera rajouté au menu en passant l'action en paramètre.

Pour finir, je l'ajoute à notre menu.

```
//Import Excel
importExcel = new ActionImporterExcel_V6(Toolkit.getDefaultToolkit().getImage(this.getClass().getClassLoader().getResource("img/excelImport.gif")));
JMenuItem importerExcel = new JMenuItem(importExcel);
menu.add(importerExcel);
```

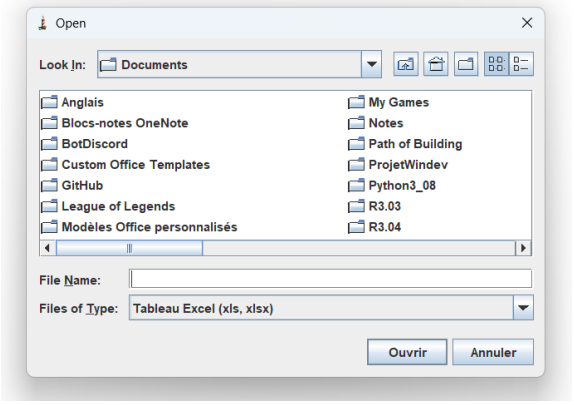
Une fois le bouton créé, je m'attaque à l'objet que je vais utiliser pour lire le fichier Excel. Cet objet sera appelé ExcelConverter.java.

Cet objet aura toutes les méthodes nécessaires pour lire le fichier Excel puis le transformer en modèle lisible par l'application ce qui nous permet de l'afficher à l'utilisateur.

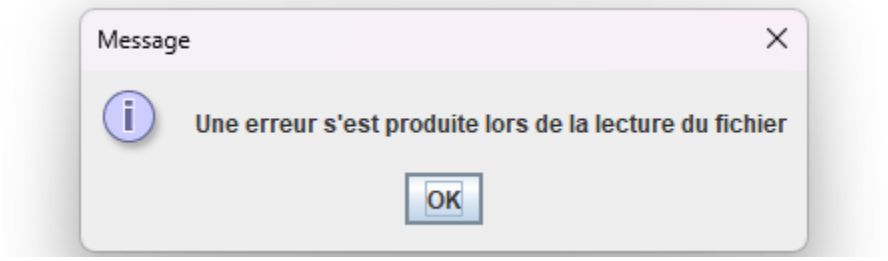
Ensuite, j'ai complété l'action précédemment créée.

L'action se déroule dans cet ordre :

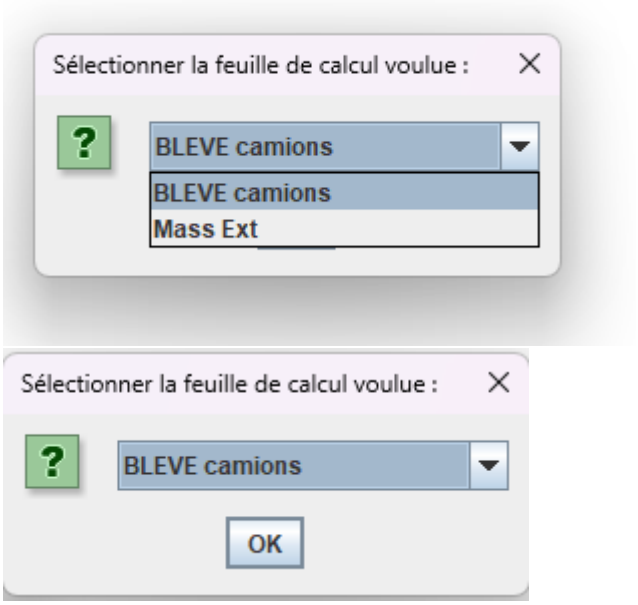
- Lors du clic sur le bouton pour importer, une page de sélection de fichier apparaît demandant uniquement des fichiers .xls ou .xlsx



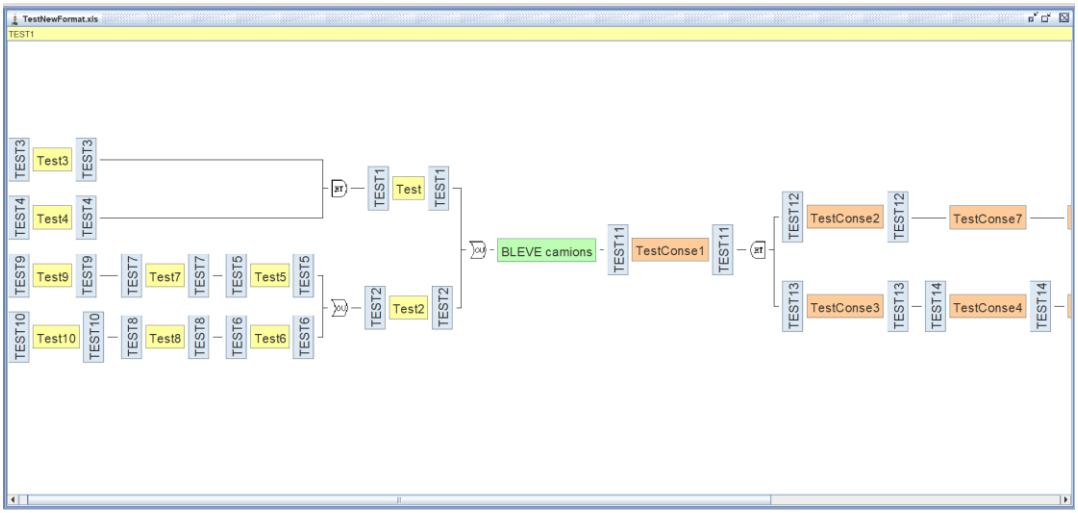
- Ensuite le fichier Excel est à sélectionner (aucune erreur possible sauf si renommage d'un fichier non .xls ou .xlsx en .xls ou .xlsx dans ce cas-là un message d'erreur disant que le fichier ne peut pas être lu sera envoyé à l'utilisateur)



- Lorsque le fichier peut être lu, un dialogue de sélection de nom de feuille est demandé à l'utilisateur (nom de la feuille de calcul voulue). Cette fenêtre est demandée par le client car il veut pouvoir mettre plusieurs nœuds dans un même fichier Excel en les séparant en feuille.



- Une fois la feuille sélectionnée, un petit temps de chargement sera nécessaire pour pouvoir charger toutes les données. Lorsque ce temps de chargement sera fini, le nœud s'affichera.

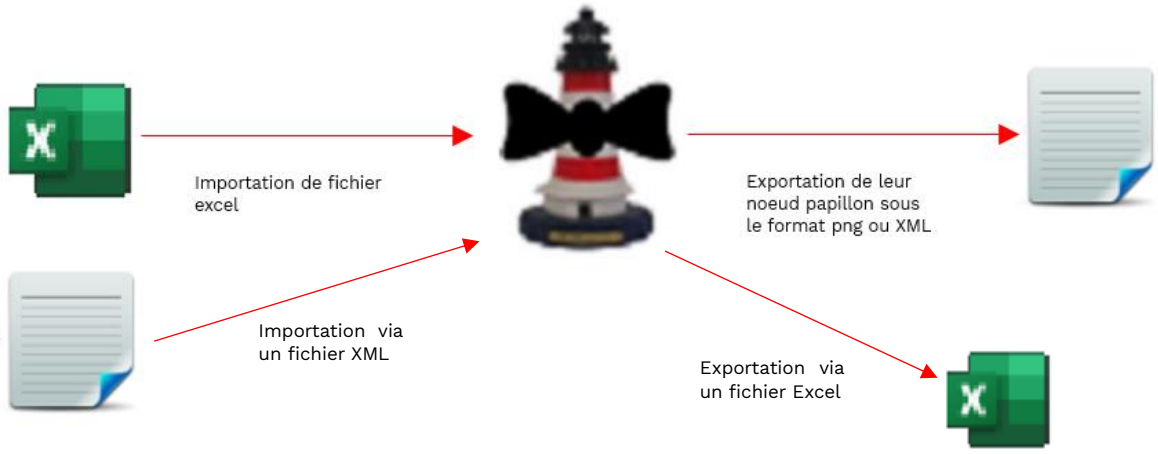


ii. Export d'un fichier Excel

Comme vu dans la partie II.1, l'export/sauvegarde d'un nœud papillon se faisait soit à la main, soit via un fichier XML. Mon travail a consisté à rajouter une option d'exportation facile via des feuilles de calcul Excel, ce qui permettrait à l'entreprise de sauvegarder les nœuds papillons créer grâce à l'application, pour pouvoir ensuite les réimporter directement dans l'application.

Chemin d'exportation d'un nœud papillon avant la mise à jour :

Chemin d'exportation d'un nœud papillon après la mise à jour :



1. Choix de la technologie à utiliser

La technologie utilisée pour l'export de fichiers Excel est la même que pour l'import.

2. Utilisation de la bibliothèques apache POI pour l'exportation

L'utilisation de la bibliothèque apache POI est la même que pour l'import mais au lieu de récupérer la Sheet et les éléments voulu à l'intérieur de celle-ci nous devons créer les lignes et insérer les données dans chaque cellule qui nous sont utiles.

Ensuite pour finir nous devons enregistrer le fichier via les FileOutputStream de Java.

3. Mise en place d'un format de fichier Excel

Le format de fichier Excel est le même que celui d'import

4. Développement de la fonctionnalité d'export

Après avoir fini d'étudier le code qui m'a été donné, de choisir la technologie que j'allais utiliser pour lire des fichiers Excel, et de mettre en place un format de fichier Excel, je commence enfin le développement de la fonctionnalité. Dans un premier temps, j'ai créé le fichier ActionImporterExcel qui correspondra à l'action qui allait se passer lors du clic sur le bouton, en laissant le fichier vide.

Ensuite, je suis passé à la création du bouton en lui-même qui a été assez simple, car plusieurs autres boutons étaient déjà présents.

Je crée mon action en lui passant en paramètre l'icône voulu.

Ensuite, je crée mon item qui sera rajouté au menu en passant l'action en paramètre.

Pour finir, je l'ajoute à notre menu.

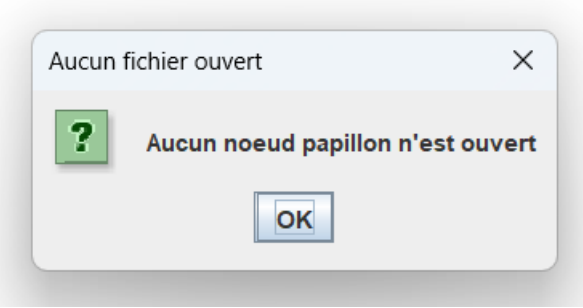
```
//Export Excel
ActionExporterExcel_V6 exportExcel = new ActionExporterExcel_V6(Toolkit.getDefaultToolkit().getImage(this.getClass().getClassLoader().getResource("img/excelExport.gif")));
JMenuItem exporterExcel = new JMenuItem(exportExcel);
menu.add(exporterExcel);
```

Une fois le bouton créé, je m'attaque à l'objet que je vais utiliser pour écrire dans le fichier Excel. Cet objet sera le même que pour la lecture et se nomme ExcelConverter.java.

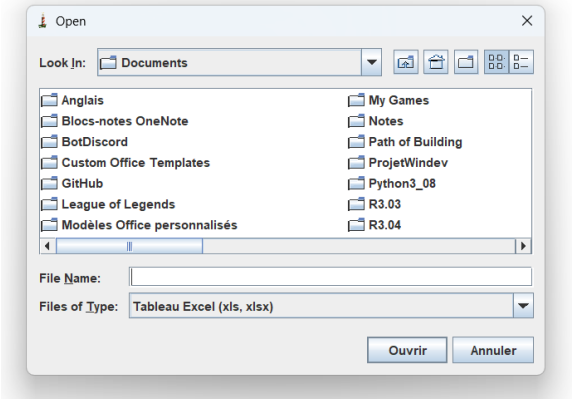
Cet objet aura toutes les méthodes nécessaires pour écrire dans le fichier Excel depuis le modèle utilisé par l'application.

L'action se déroule dans cet ordre :

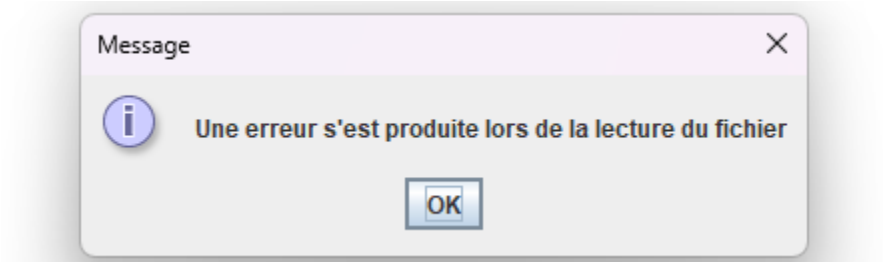
- Dans un premier temps nous devons avoir un nœud papillon d'ouvert si jamais l'utilisateur venais à cliquer sur le bouton pour exporter sans nœud papillon d'ouvert un message d'erreur apparaîtra.



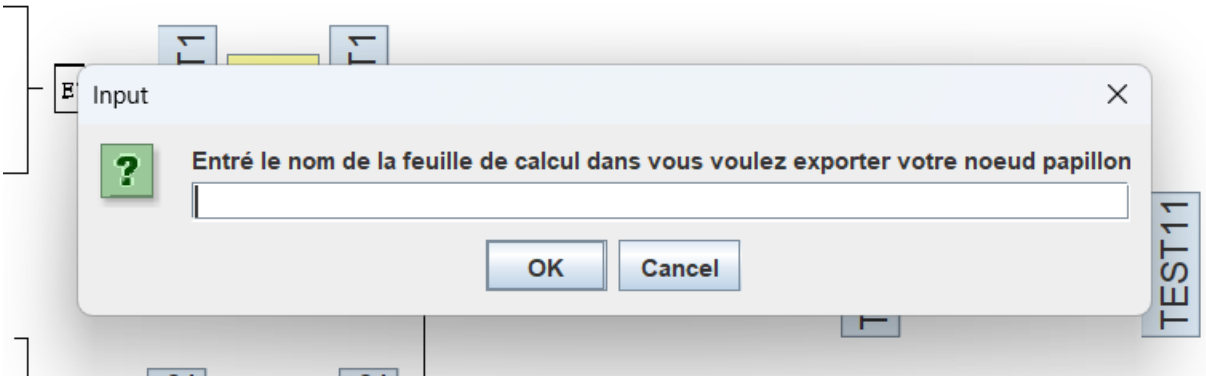
- Lors du clic sur le bouton pour exporter, une page de sélection de fichier apparaît demandant uniquement des fichiers .xls ou .xlsx



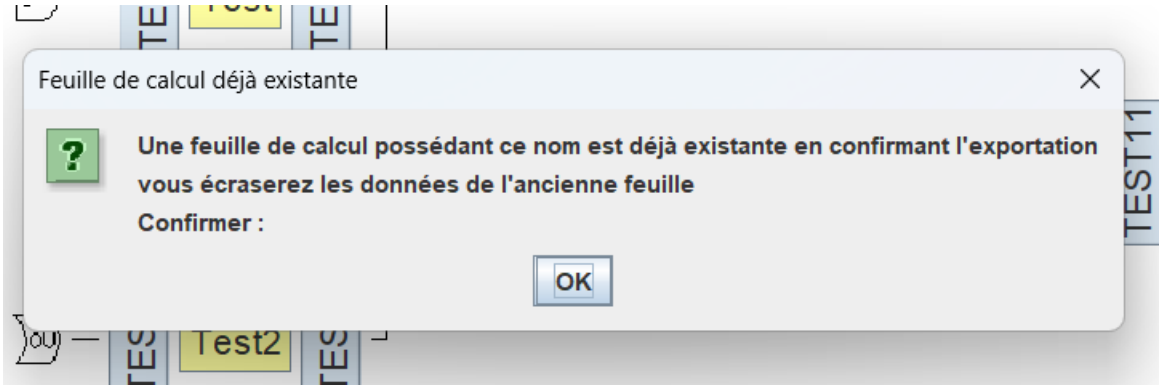
- Ensuite le fichier Excel est à sélectionner (aucune erreur possible sauf si renommage d'un fichier non .xls ou .xlsx en .xls ou .xlsx dans ce cas-là un message d'erreur disant que le fichier ne peut pas être lu sera envoyé à l'utilisateur)



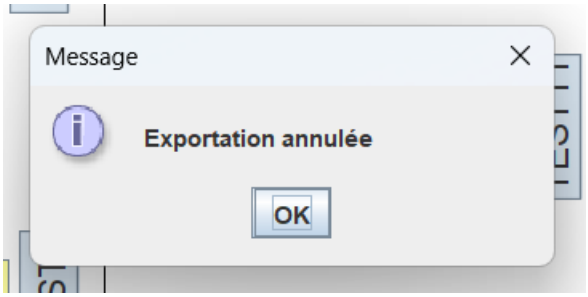
- Si un fichier Excel valide est sélectionné par l'utilisateur alors dans ce cas-là une prompt demandant le nom de la feuille de calcul dans laquelle on veut sauvegarder notre nœud papillon.



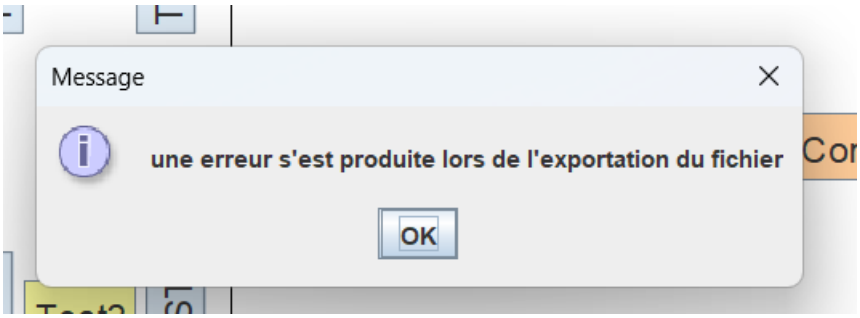
- Si jamais l'utilisateur entre le nom d'une feuille de calcul déjà existante alors un message demandant de confirmer l'exportation apparaît.



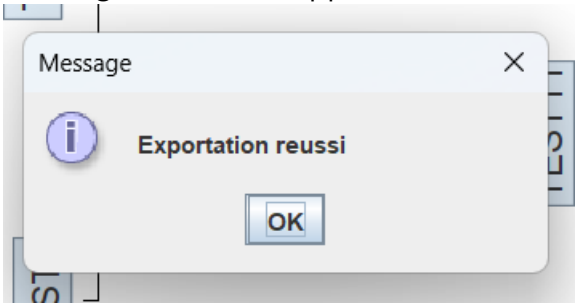
- Si l'utilisateur annule l'action en cliquant sur la croix alors un message confirmant l'annulation de l'exportation est afficher



- Une autre erreur peut arriver si jamais l'utilisateur à le fichier dans lequel il veut exporter son nœud d'ouvert alors ce message apparaît



- Si l'utilisateur confirme ou met un nom d'une feuille de calcul qui n'existe pas et que le fichier n'est pas ouvert alors l'exportation seras un succès est un message de succès apparaîtra



IV. Bilan

I. Sur le plan technique

Le sujet proposé par ce stage est l'ajout de fonctionnalités sur un logiciel développé et maintenu par différents développeurs. L'ajout de modifications au fil des versions a donné un projet assez conséquent, et l'utilisation de méthode d'analyse efficace était indispensable pour s'y retrouver. J'ai alors appris à utiliser de nombreuses fonctionnalités de l'environnement de développement Eclipse pour me faciliter la tâche.

Tout d'abord, j'ai appris à lire et essayer de comprendre un code existant. La documentation fournie par les anciens développeurs ayant travaillé sur le logiciel m'a grandement aidé dans la compréhension du code. Toutefois, une partie du code n'étant pas documentée, j'ai dû l'analyser pour saisir l'ensemble des subtilités du logiciel. La maîtrise totale du code s'est faite au fur et à mesure de mon avancement.

Ensuite, j'ai utilisé de nombreuses fonctionnalités d'Eclipse, notamment celle permettant d'afficher une hiérarchie des types d'une classe ou encore une hiérarchie d'appels d'une méthode ou d'un attribut.

Dans certains cas, j'ai dû avoir recours au refactoring, un procédé permettant de faire des modifications sans que cela ait un impact sur le fonctionnement normal du logiciel. Eclipse propose aussi plusieurs fonctions pour réaliser le refactoring de manière efficace.

La documentation étant très importante dans un projet qui risque d'être repris par de futurs programmeurs, j'ai appris à utiliser la Javadoc efficacement et à fournir une documentation la plus complète possible. J'ai aussi mis en forme la Javadoc en utilisant des balises html, des liens hypertextes redirigeant vers la Javadoc d'autres classes/méthodes, mais aussi des « tags » qui ajoutent des compléments sur la Javadoc (comme l'auteur, la date, si la méthode est dépréciée…). Ainsi, grâce à ce travail, les futurs programmeurs pourront comprendre facilement et rapidement le code du logiciel SSAM.

Ce stage m'a permis d'approfondir mes connaissances dans le langage Java, de mieux comprendre son fonctionnement.

Modifier un projet existant pouvant être délicat, il est nécessaire de respecter le plus possible l'architecture de base sans pour autant complexifier le code. J'ai donc appris à mettre en place de bonnes pratiques de programmation (garder des classes simples, éviter les dépendances entre classes, utiliser les commentaires à bon escient…).

II. Sur le plan humain

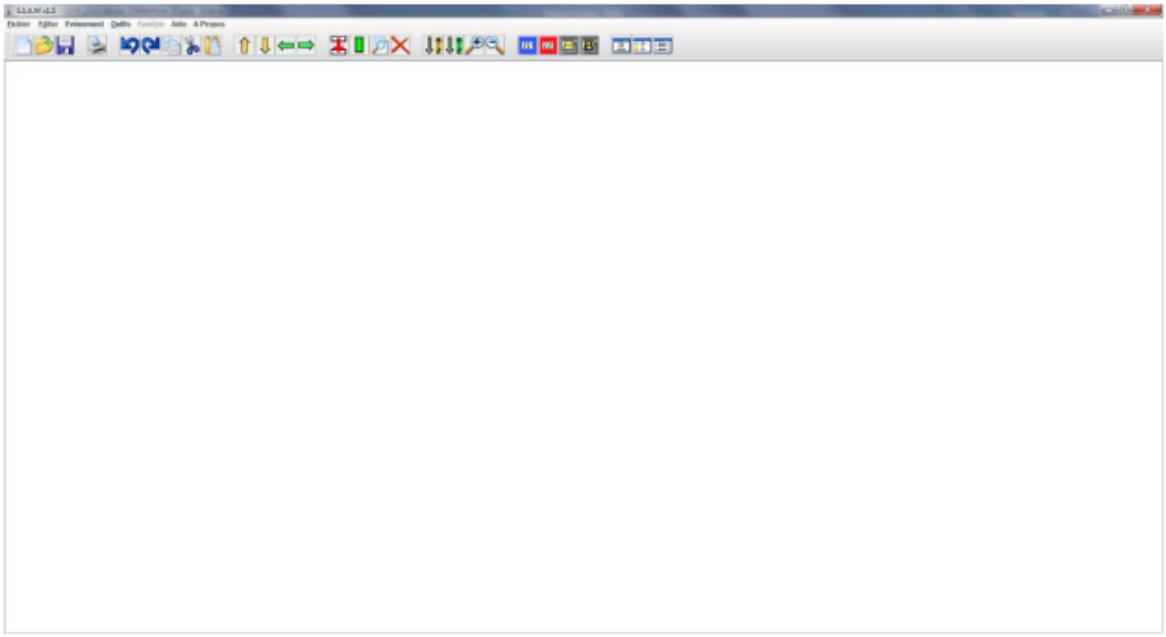
Au cours de ce stage, des besoins client ont été formulés et modifiés. Ces besoins m'ont appris à prendre des notes, les trier puis pouvoir proposer plusieurs solutions au client pour répondre à ses besoins.

V. Conclusion

Ce stage a été très riche sur le plan technique principalement. Travailler sur un projet existant depuis longtemps et qui a eu plusieurs développeurs différents m'a permis d'améliorer mes méthodes d'analyse. La tâche principale ayant pu être réalisée, plusieurs autres tâches m'ont été données. Certaines ont dû être revues à cause de la contrainte de temps, voire abandonnées.

De futures améliorations sont à prévoir que ce soit au niveau graphique comme au niveau pratique.

Annexe 1



Fenêtre principale de l'outil SSAM 1.5.