Thomas **DE SIANO**
Cédric **ESCALLIER**
Thierry **BAUD**

esil
département informatique
3ième année

Projet de modélisation orientée objet

Rational

Professeur
Mr. F. **DUMAS**

Table des matières

Chapitre 1 : Etude sur Rational ROSE Entreprise v.2001A

Rational reste le premier vendeur d'outils de modélisation UML avec ROSE.

Afin de mieux apprécier ce logiciel, ses possibilités et limitations, nous avons décidé d'utiliser celui-ci dans le cadre du projet de Modélisation Orientée Objet.

Les fonctionnalités, la stabilité, les possibilités en font un logiciel à la fois complet et très apprécié des utilisateurs.

Nous allons voir en détail le logiciel, ses composants, et allons exposer notre avis sur celui-ci.

1.1. Fonctionnalités

Le but premier d'un AGL (Atelier de Génie Logiciel) est la conception. Cette phase est essentielle pour le développement contrôlé et géré d'un logiciel. Ainsi, l'AGL doit nous donner accès rapidement et simplement aux outils pour la modélisation du produit.

La documentation technique de Rational ROSE donne une liste exhaustive des fonctionnalités de son produit.

Il serait inutile de détailler ici cette liste complète, mieux vaut en donner les aspects les plus intéressants – utilisés.

Tout d'abord, la principale utilisation des outils AGL : Les diagrammes : se sont eux qui déterminent le système et donnent la vue d'ensemble, l'enchaînement des actions, …

1.1.1. Les diagrammes

Rational ROSE intègre complètement les diagrammes :

- ✓ Diagramme Use Case
- ✓ Diagramme de classe
- ✓ Diagramme de séquence, collaboration, activité
- ✓ Diagramme de composant, déploiement

Sur le marché, il est de loin celui qui a su intégrer tous ces diagrammes (avec quelques petits problèmes négligeables – cela sera traité dans la partie problème)

1.1.1.1. Gestion de versions

De base, Rose n'intègre pas la gestion des versions ; mais en rajoutant un plugin, celui-ci gère les versions (par exemple avec CVS).

1.1.1.2. Exportation

Rose permet l'exportation de fichiers vers Word. Ceci à son avantage et son désavantage : d'autres utilitaires permettent l'exportation vers des fichiers images, GIF, qui est beaucoup plus pratique et intégrable à des documents.

1.1.1.3. Production de code

Sans conteste, un point fort pour Rational : Il permet la génération de code C, C++, Java Visual Basic, ADA, Corba, XML, JSP.

Le code généré est « agréable », quoiqu'un peu moins lisible qu'avec le logiciel Together.

Il produit aussi du code « templates ».

1.1.1.4. Autres

Le Reverse Engineering est également supporté par Rose ; néanmoins il ne faut pas en abuser car est limité et seulement utile pour la compréhension de la structure générale d'un système objet, pas plus.

Rational Rose est de plus un logiciel polyvalent, dans le sens où il est utilisable sur différents systèmes d'exploitation, en utilisant des ressources 'modestes' :

- Windows :
  - Microsoft Windows NT 4.0 SP6
  - Windows 98
  - Windows 2000 SP1
- Matériel nécessaire :
  - Microsoft Windows NT 4.0 SP6, Windows 95, Windows 98, ou Windows 2000
  - PC-compatible
  - 64 MB of RAM (128 MB recommandé)
  - 200 MB d'espace disque
  - Carte vidéocompatible
- UNIX/Linux
  - Linux x86 (Red Hat 6.2, 7.0)
  - Solaris 2.5.1, 2.6, 7, 8
  - HP-UX 10.20, 11.0, 11.i
  - SGI IRIX 6.5.5
- Matériel nécessaire
  - Mémoire : 64 MB plus 32 MB of RAM par utilisateur
  - Vidéo : X Windows
  - Espace disque : 270 MB
  - Media: CD-ROM

1.2. Ergonomie

Rose est un logiciel très fonctionnel mais avant tout il sait ravir l'utilisateur par une interface très intuitive.

Les divers menus déroulants, raccourcis, clics droits donnent accès rapidement aux propriétés et aux modifications possibles.

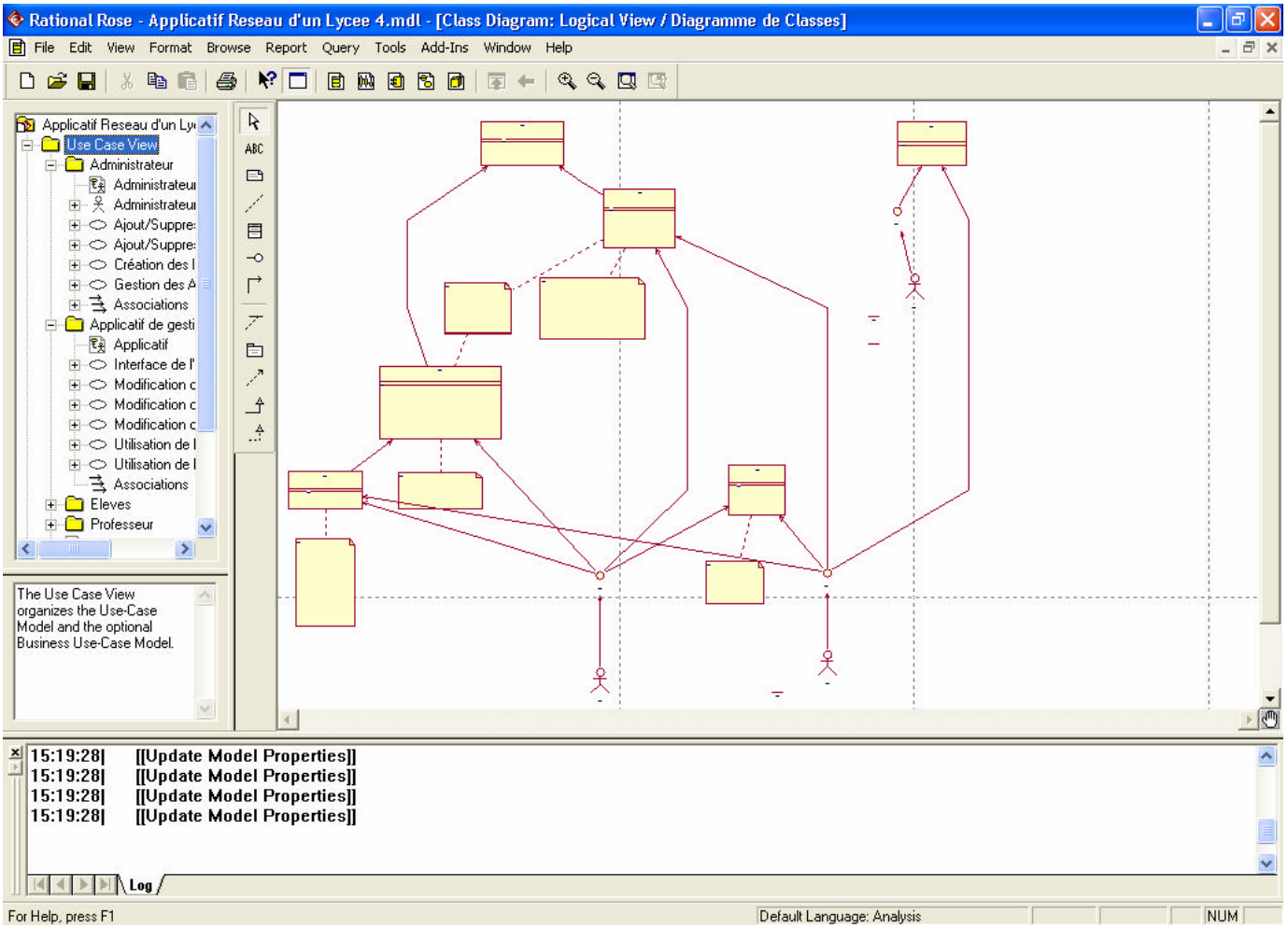
L'interface qui serait qualifiable de classique dans un environnement Windows, permet de se déplacer facilement entre les menus, diagrammes.

Un aspect d'autant plus intéressant que pratique est l'apparition d'un menu particulier, fonction de l'objet sélectionné.

Les couleurs est un plus : sur chaque lien réalisé, chaque objet, il est possible de colorier celui-ci d'une couleur donnée : cela évite de se perdre dans les schémas.

Enfin, pour les schémas de grande taille, des pointillés viennent se rajouter au dit schéma pour voir quel sera le découpage à l'impression.

Voici un exemple de l'interface où l'on retrouve les points ci-dessus :



1.3. Points Forts

Rose est un logiciel très puissant, les points forts le concernant sont nombreux et les impressions à l'utilisation vont dans le même sens. Nous allons voir point par point quels sont les avantages de Rose :

1.3.1. L'interface

Comme vu précédemment, l'interface graphique de Rose est à la fois très conviviale mais aussi très riche. L'utilisateur va pouvoir avoir accès rapidement aux fonctionnalités voulues.

1.3.2. Puissance

A quelques limitations près (dues à l'implémentation d'UML dans Rose), Rose se montre comme un logiciel puissant et robuste. Celui-ci n'a jamais « planté » quelle qu'en ai été l'utilisation.

1.3.3. Implémentation de UML

On peut dire que Rose est l'AGL qui implémente le mieux l'UML. Certes, il n'implémente pas l'UML à 100% mais celui-ci est très complet et permet de réaliser de façon rapide la modélisation d'un projet.

1.3.4. Lisibilité

Il est clair que Rose propose tout un tas de fonctionnalités permettant de rendre clair les diagrammes : couleurs, objets différents, …

1.3.5. Reverse Engineering

Même si cela n'a pas été traité par le sujet proposé, il faut savoir que Rose est le seul AGL à implémenter le reverse engineering.

Note : définition de reverse engineering :

En général, il s'agit de décompiler ou de désassembler un programme, c'est-à-dire de prendre quelque chose de compréhensible par une machine mais pas par un être humain et d'en faire quelque chose de lisible pour un homme mais plus pour un ordinateur.

Dans la pratique, le reverse engineering consiste à analyser le fonctionnement d'un programme ou d'un périphérique (typiquement en analysant la communication qu'il a avec le reste du système), ou alors en examinant pas à pas les résultats de son exécution.

1.3.6. Prix

Apparemment, la licence d'utilisation de Rose ne coûterait environ que 8000 à 9000 FF (1300 €) ; Quand on pense au temps gagné à utiliser un tel logiciel, il semble évident que l'investissement en vaut le coût.

1.4. Points Faibles

Comme tout logiciel, Rose a beaucoup d'avantages ; il convient néanmoins d'analyser les éléments qui nous ont arrêtés pour la modélisation des projets avec cet outil.

1.4.1. Supports linguistiques

Rational Rose n'est distribué qu'en version anglaise. Cela peut arrêter des utilisateurs n'ayant pas une connaissance technique de l'anglais.

1.4.2. Richesse du programme

Rose est tellement complet qu'il arrive parfois que l'on soit « submergé » d'informations, dont la plupart ne nous sera jamais utile. Il faudrait une version « light » utilisable pour des réalisations rapides, en épurant les menus, ...

1.4.3. Problèmes pour la réalisation

Là encore, nous avons expérimentés quelques problèmes pour la modélisation des problèmes :

1.4.3.1. Relations Package – Use Case

Voici le problème : Un package a été réalisé. Afin d'intégrer celui-ci dans le diagramme et pouvoir en tirer toutes les fonctionnalités directement, il faudrait pouvoir faire un lien entre le dit package et l'objet. Cela est impossible à réaliser avec Rose.

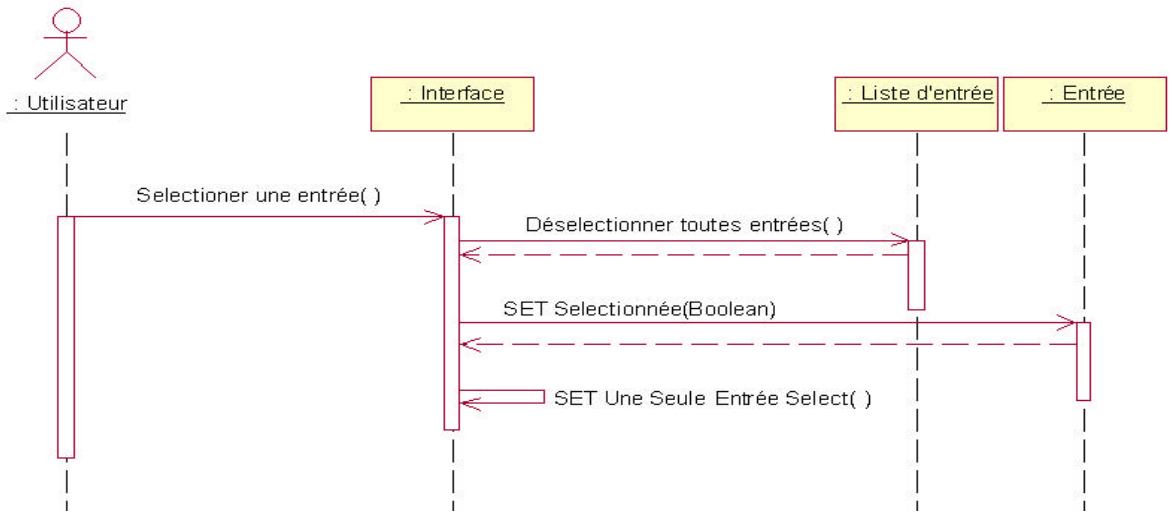
Il faut donc dans le Use Case faire une nouvelle tâche avec le nom du package, lié à l'objet.

Mais avec cette solution on perd tout l'avantage du package...

1.4.3.2. Diagrammes de séquence

Là encore Rose pose un problème quand à la réalisation de diagrammes de séquence :

En effet, par exemple, des évènements externes (clic souris, ...) vont toucher l'interface : il FAUT qu'un objet soit associé à l'évènement, il ne peut y avoir une action ne venant d'aucun objet.



1.4.4. Conclusion

En fin de compte, Rose est un logiciel très puissant qui sait satisfaire chacune de nos envies. Malgré quelques inconvénients à son utilisation, Rose reste très complet.

Il fut très agréable de travailler sur un logiciel d'une telle qualité.

L'étape suivante consisterait à tester d'autres logiciels de modélisation afin de mieux apprécier la puissance de Rose, et pouvoir se faire une idée plus objective au vu des avantages et inconvénients.

Chapitre 2 : Applicatif réseau d'un lycée

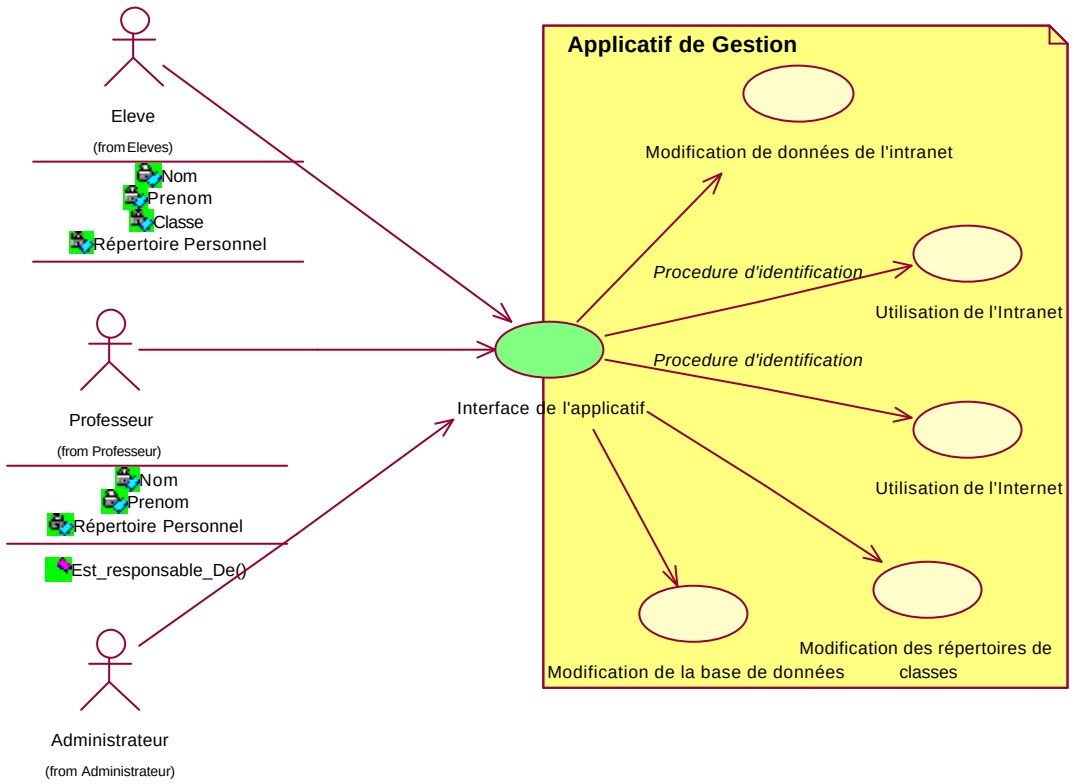
C'est après une étude détaillée du cahier des charges que ce dossier utilisateur a été établi. Il a pour vocation de répondre aux besoins d'un lycée en termes de gestion de son réseau. Afin de faciliter sa compréhension par tous, nous avons découpé cette étude en plusieurs parties.

Nous présenterons tout d'abord une vue de Use Case, c'est-à-dire un ensemble de diagrammes permettant d'obtenir une vision globale de l'applicatif à développer.

Nous développerons ensuite une vue logique du système afin d'envisager la mise en place de notre solution.

2.1. Vue de Use Case

Afin de mieux répondre aux attentes du lycée, nous avons structuré l'applicatif de gestion en différentes parties :



Notre vision s'appuie donc sur plusieurs visions d'une même application. En effet, notre applicatif ne doit pas offrir des possibilités équivalentes à plusieurs types d'utilisateurs. Nous nommerons ces personnes des acteurs dans le cadre de cette étude.

Nous allons maintenant décrire plus précisément les possibilités des différents types d'acteurs de notre système grâce à des diagrammes de Use Case distincts.

2.1.1. Use Case 'Élèves'

Il s'agit ici de présenter les différentes opérations pouvant être effectuées par un élève du lycée. (cf. Annexe 1.1 ; Use Case Élèves)

Tout élève du lycée pourra donc utiliser les ressources mises à sa disposition après s'être identifié auprès de la base de donnés. Si toutefois un élève n'avait pas encore de compte sur la base de données, il devrait consulter un enseignant responsable de sa classe afin d'obtenir une possibilité d'inscription.

2.1.2. Use Case 'Professeur'

Ce diagramme de Use Case permet de mieux cibler les différentes possibilités offertes aux professeurs. Pour faciliter la lisibilité, nous avons regroupé les différentes activités en ensembles. (cf. Annexe 1.2 Use Case Professeur)

Pour chacune d'entre elles, nous avons placé en commentaire du diagramme leurs principales

La possibilité de restauration des systèmes endommagés est offerte aux professeurs. Pour cela, il leur suffit d'aller récupérer l'image de ce système dans une armoire contenant l'ensemble des sauvegardes. Ces sauvegardes sont disponibles sous la forme de Cd-rom bootables (démarrage automatiques) qui exécutent un script afin de permettre la restauration rapide et automatisée du système. À la fin de l'opération, il suffit de replacer l'image dans le classeur adéquat.

2.1.3. Use Case 'Administrateur'

Ici encore, nous cherchons à représenter, de manière lisible, l'ensemble des possibilités de l'administrateur. (cf. Annexe 1.3 Use Case Administrateur)

La démarche suivie est également identique aux précédents Use Case, c'est-à-dire que nous essayons de regrouper les activités par type afin de donner une meilleure vision des possibilités de notre applicatif.

Au niveau de la restauration de systèmes, le principe reste identique à celui mis en œuvre dans le Use Case Professeur. Par contre, la création d'images du système n'est effectuée que par l'administrateur. Il dispose de plusieurs moyens, en fonction du système utilisé.

Toute la gestion de ces images se fera grâce à un logiciel de clonage de disque dur. En effet, cela nous permet de conserver des images simples à mettre en place et nous ouvre plusieurs possibilités en termes de sauvegarde : Elle peut être effectuée en ajoutant un disque dur (temporaire) sur la machine locale ou via le réseau si les pilotes du matériel réseau sont disponibles. Dans tous les cas, l'administrateur n'aura qu'à graver l'image du système sur un Cd-rom type qui comporte un secteur d'amorce pour la machine et un script de restauration.

Signalons également qu'il conviendra de verrouiller les BIOS des systèmes afin de limiter les restaurations non autorisées.

2.1.4. Diagrammes d'activité & Fiches Use Case

2.1.4.1. Présentation des diagrammes d'activité

En s'intéressant ici plus aux actions qu'aux états, ils montrent l'activité et le fonctionnement d'une opération d'une classe, par exemple.

2.1.4.2. But

Ceux-ci présentent le séquencement des activités du projet.

Les fiches de Use Case (annexe 1.4) complètent ces descriptions au cas par cas (Annexes 1.5, 1.6, 1.7).

Ils font apparaître des notions de l'applicatif comme l'identification des utilisateurs. En effet, c'est ici qu'on se rend pour la première fois compte de ces limitations liées aux droits des utilisateurs. En comparant les 3 diagrammes précédents, on peut observer que les possibilités offertes sur la gestion des données sont de plus en plus importantes de l'élève jusqu'à l'administrateur.

2.1.5. Diagramme de Classe

2.1.5.1. Définition des Diagrammes de Classe

Le diagramme de classes est le diagramme le plus important de la modélisation orientée objet. Il saisit la structure statique d'un système, en montrant un ensemble de classes, d'interfaces et de collaborations, ainsi que leurs relations.
Il fait abstraction des aspects dynamiques et temporels. Il constitue le cadre dans lequel les diagrammes décrivant les aspects dynamiques du système s'insèrent.

2.1.5.2. But

Il s'agit ici de donner une direction générale pour implémenter cet applicatif de manière proche à notre vision (Cf Annexe 1.8). Un développeur se confrontant à la réalisation d'un tel système pourrait s'appuyer sur ce modèle afin de simplifier sa phase de réflexion.

Chapitre 3 : Application de gestions de password

3.1. Introduction

Le but de cette partie était de réaliser un dossier de conception en vue d'un développement d'une application de gestions de password. Pour constituer ce dossier nous avons aussi opté pour l'AGL de Rationnal: ROSE. Ce dossier est présenté dans l''annexe 2 et comporte:

- Une vue du use case pour l'application de gestions de password. Cette vue nous a permis de traduire le cahier des charges sous forme d'un seul diagramme regroupant les fonctionnalités de l'application, mais aussi l'ensemble des actions réalisable par l'utilisateur.
- Trois diagrammes de classe, qui permettent de voir en détail la structure du programme.
- Trois diagrammes de séquence, qui montrent plus précisément le traitement des interactions utilisateur. C'est-à-dire que l'on s'intérresse ici à la dynamique du programme.

3.2. Modèle statique [cf. Annexe 2 – Les diagrammes de classe]

Notre application comporte trois grandes classes :

- **La classe Interface** qui regroupe la gestion de l'interface utilisateur. Elle gère l'affichage des fenêtres et des menus, mais aussi les interactions utilisateurs.
- **La classe Cryptage** qui regroupe les méthodes pour crypter et décrypter un fichier. Cette classe n'est pas un package indépendant de l'application, elle contiend toutes les données nescessaire pour ce cryptage:
    - Le chemin du fichier source.
    - La liste des entrées.
    - La clé de cryptage.
- **La classe Liste d'entrée** qui est en fait une agglomération de classe entrée. Cette classe offre de plus toutes une pannoplie d'opérations pour la gestion de listes.

3.3. Modèle Dynamique [cf. Annexe 2 – Les diagrammes de séquences]

Après avoir vue l'architecture du programme nous nous sommes penché sur le déroulement de celui-ci. Ceci est donc passé par le traitement des événements utilisateurs. Nous avons regroupé ces événements selon leur apparition:

- dans le menu fichier : Diagramme Séquence sur le menu Fichier.
    - Ouvrir un fichier
    - Créer un nouveau fichier
    - Fermer le fichier
    - Quitter
- dans le menu Action: Diagramme Séquence sur le menu Action.
    - Modifier
    - Afficher
    - Trier
    - Rechercher
    - Créer nouvelle entrée
    - Supprimer entrées selectionnées

- Evénement Souris : Diagramme Séquence gestion souris
    - Selection d'une entrée.
    - Double-clic sur une entrée.
    - Selection de plusieurs entrées.

Après avoir vu les différentes contraintes techniques que cette application générait nous avons pu faire des choix concret sur l'implentation proprement dite du logiciel.

3.4. Choix techniques

Notre application doit fournir le maximum de fonctionnalité pour l'utilisateur mais elle doit être "légère" et rapide pour pouvoir être lancée souvent et rester en tâche de fond en attendant une prochaine utilisation. Pour cela, L'application devra avoir une interface minimaliste et contrairement au idées reçues il n'y aura pas de gestion d'utilisateurs au démarrage de l'application. La gestion des utilisateurs se fera avec un nom de fichier et sa clé d'encryption. De plus un utilisateur pourra posséder plusieurs fichiers de passwords.

En ce qui concerne l'implentation, nous avons choisi de développer l'application en JAVA avec un outil d'aide pour le développement des interfaces Homme Machine. Ceci permet à l'application d'être indépendante du système d'exploitation et surtout permet d'être lancée à partir d'un navigateur internet. (cf. Diagramme de composant)

De plus la modélisation avec UML permet d'avoir une vision orientée objet très proche voir identique avec la vision des objets en JAVA. Ceci nous a permis de nous focaliser sur la conception et de ne pas nous "étallé" sur les moyens utilisé pour l'implémentation comme on l'aurait fait si on avait développé avec Visual Basic.

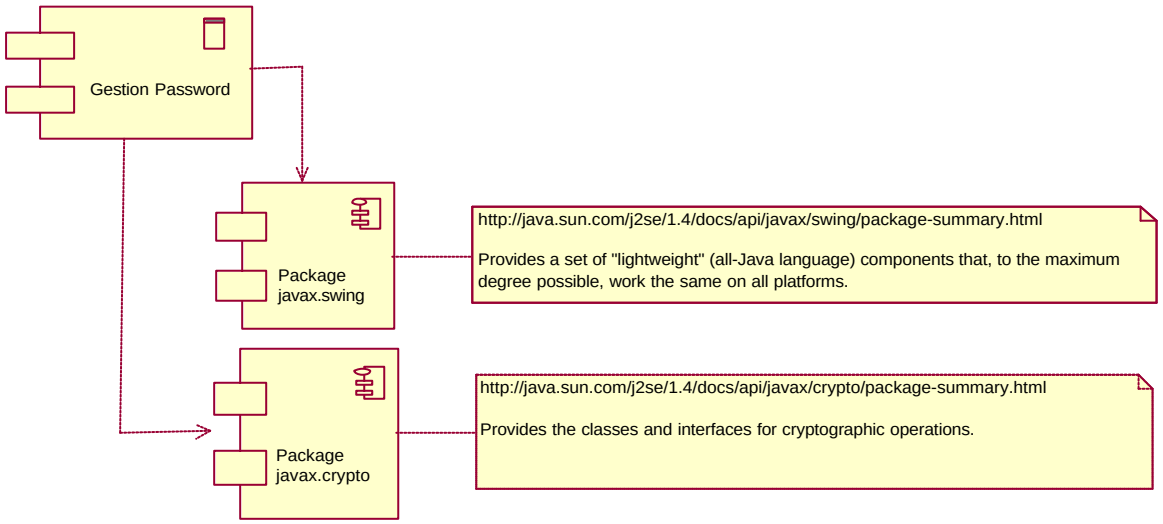


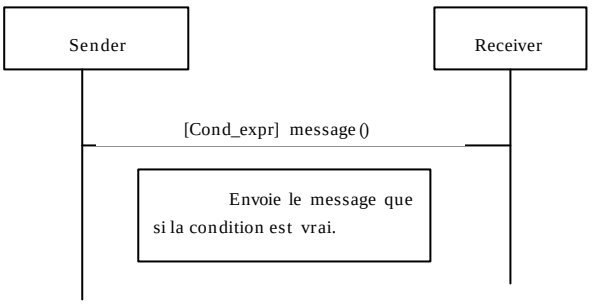
Diagramme de composant.

3.5. Problèmes rencontrés

Les problèmes rencontrés les plus important était surtout au point de vue de la prise en main d'un AGL. Ceci est du au fait que les AGL fournit sur le marché ne permettent pas encore de représenté la totalité des diagrammes spécifié par la norme UML.

Par exemple, pour représenter, dans un diagramme de séquence, une condition.

En UML:



Alors que dans les AGL ont ne peut pas spécifier de condition, il faut donc rajouter des commentaires pour représenter une condition.

3.6. Conclusion

Ce projet nous a permis d'utiliser un AGL, et de réaliser toute la phase conception pour le développement d'une application.

Les futures évolutions de cette application pourraient être de pouvoir gérer des services WEB qui nécessitent un mot de passe sans que l'utilisateur n'ait à faire de copier/coller dans les champs. C'est-à-dire, une fois l'application lancée, elle offre le choix de cliquer sur l'URL et d'identifier l'utilisateur automatiquement.

Conclusion

Ce projet a été pour nous l'occasion de mettre en pratique le cours de modélisation orientée objet. Ce fut un projet à la fois intéressant par son sujet et sa difficulté mais aussi formateur grâce à l'utilisation de Rationnal® Rose.

Bien que nous n'ayons pas exploité toutes les fonctionnalités du logiciel nous avons particulièrement apprécié l'utilisation de ce logiciel.

Il aurait été peut-être intéressant d'aller jusqu'à la génération de code afin de voir la puissance du logiciel.

Documents annexes 1

Documents annexes 2