Rapport de projet de LO43
Université de Technologie de Belfort-Montbeliard

Affichage de problématiques d'habillages liées au transports urbain

Rapport de projet

Julien VOISIN
Guillaume OBERLE

Belfort, le 14 juin 2013

Table des matières

Introduction

Dans le cadre de l'unité de valeur LO43, il a été demandé aux étudiants de réaliser un projet de fin de semestre. Ce projet a pour but de mettre en application les différentes notions acquises en programmation orientée objet ainsi que de pratiquer le langage de programmation Java.

L'objectif de ce projet est de réaliser une application fonctionnelle afin d'évaluer et de représenter dans une interface graphique un couple de données/solution du problème d'habillage dans le cadre de la planification des transports de bus urbains.

Dans une première partie nous présenterons le cahier des charges. Dans une deuxième partie nous ferons l'analyse UML de notre projet. Dans une troisième partie, nous évoquerons la façon dont nous avons choisi d'implémenter notre solution. Puis nous ferons état des choix techniques que nous avons du réaliser. Enfin, nous terminerons par une conclusion.

1 Cahier des charges

1.1 Présentation de la famille du problème

Le problème de *crew scheduling* (*problème d'habillage* en français) est un problème d'optimisation connu, avec de multiples applications dans la vie réelle, telles que des probatiques de répartition d'équipes en entreprise, l'organisation de tournées, . . .La problématique soulevée pour ce projet est une répartition au mieux des taches de conduites en cherchant à minimiser le nombre de chauffeurs afin de maximiser les profits.

1.2 Objectif

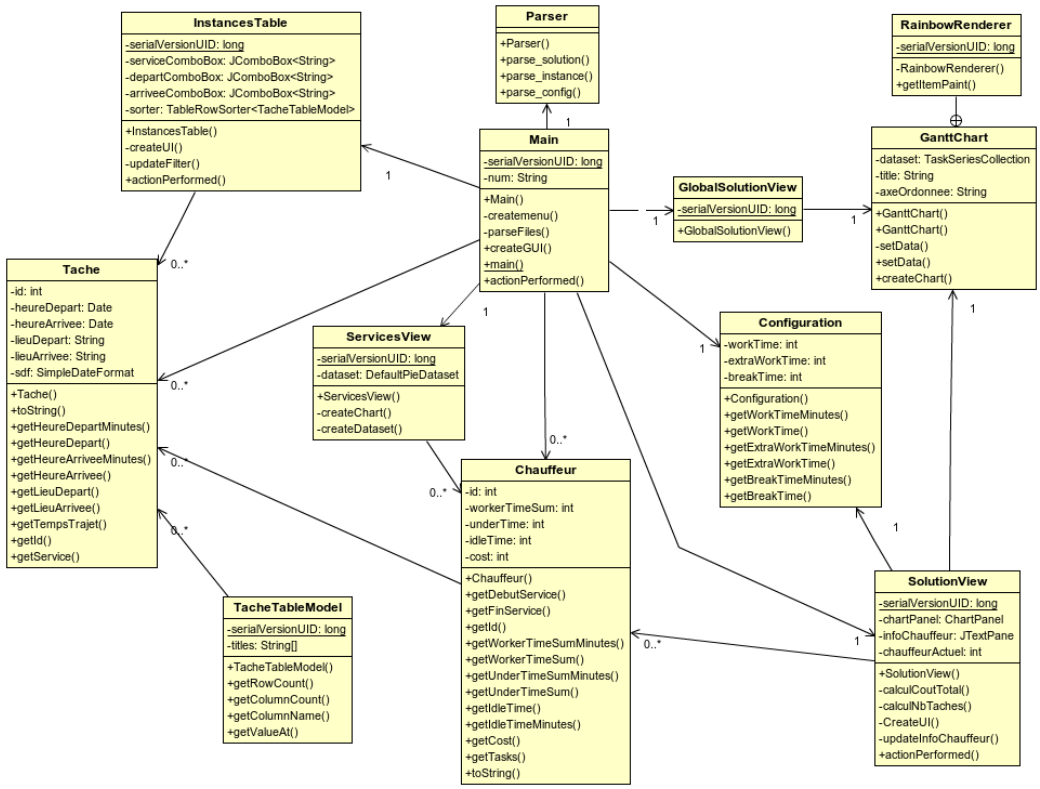
L'objectif de ce projet est d'évaluer et de représenter dans une interface graphique une solution du problème, afin de la contextualiser et d'en permette une visualisation rapide et efficace.

1.3 Données

Pour ce projet, on dispose à la fois des données du problème et d'une solution associée. La donnée du problème est un fichier décrivant toutes les tâches de conduites. Chaque tâche est composée par quatre éléments : l'heure de départ, l'heure d'arrivée, le lieu de départ et le lieu d'arrivée. Des fichiers Donnée et Solution sont fournies dans un format précis.

2 Analyse UML et contraintes techniques

2.1 Analyse UML



Ce diagramme UML représente l'intégralité du projet. Il est amusant de noter qu'il ne comporte que des associations, et une agrégation, et aucun héritage.

2.2 Contraintes techniques

- Le rapport, ainsi que le projet doivent être rendus pour le vendredi 28 juin 2013.
- Le projet doit être codé en Java
- L'interface doit être graphique

3 Implémentation

L'implémentation peut se découper en 4 parties :
- Parsage des fichiers
  - de solutions
  - d'instance
  - de configuration
- Affichage des instances
- Affichage du rapport des services
- Affichage de la solution par chauffeur
- Affichage global de la solution

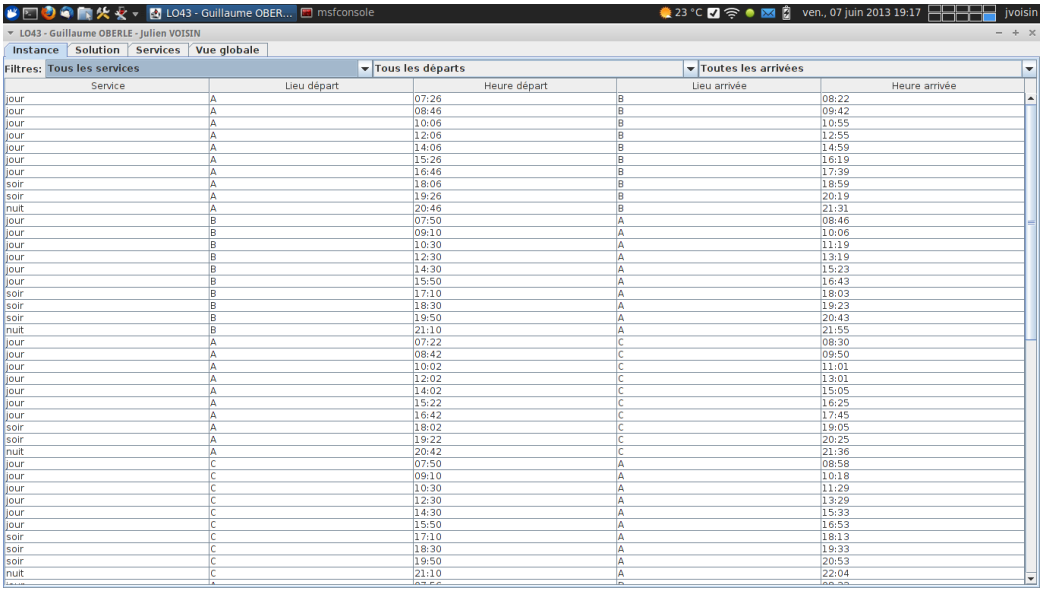
3.1 Parsage des données

Le jeu de données se passe en paramètre au programme. Des valeurs par défaut son appliquées sinon. Un fichier de solution parsé retourne une liste de tâches, une fichier de configuration, une classe *Config*, et enfin un fichier de solutions retourne une liste de chauffeurs. La classe dédiée a cette tâche ne comporte qu'une petite centaine de lignes.

3.2 Affichage

La partie graphique du projet utilise *SWING*, ainsi que la bibliothèque *JfreeCharts* (voir plus loin). La fenêtre est composée d'onglets permettant de sélectionner la vue. Chaque onglet menant à une vue, correspondant chacune à une classe héritant de *JPanel*.

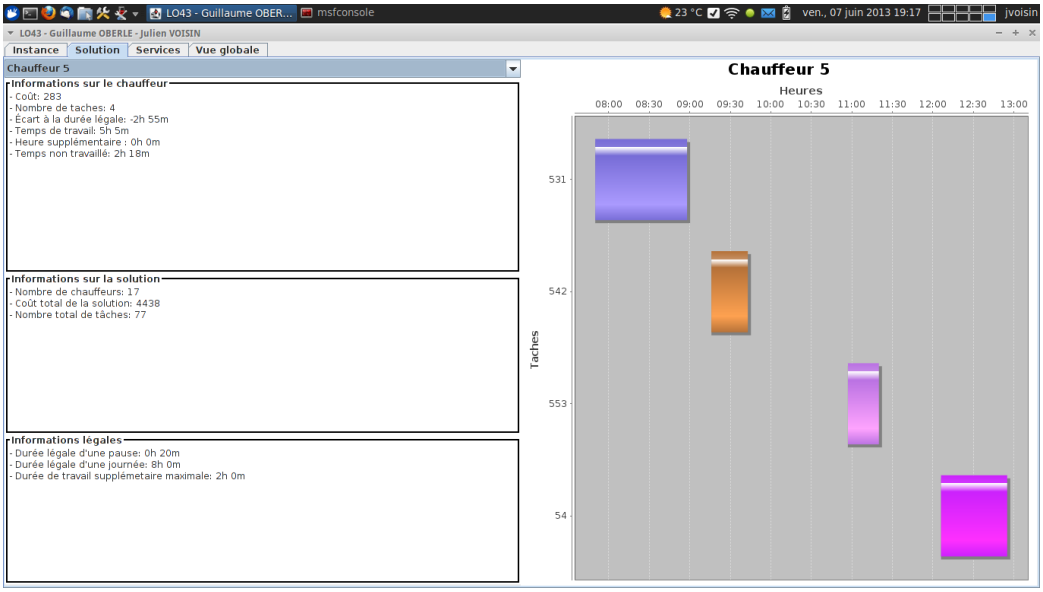
3.2.1 La vue *Instances*



Service	Lieu départ	Heure départ	Lieu arrivée	Heure arrivée
jour	A	07:26	B	08:22
jour	A	08:46	B	09:42
jour	A	10:06	B	10:55
jour	A	12:06	B	12:55
jour	A	14:06	B	14:59
jour	A	15:26	B	16:19
jour	A	16:46	B	17:39
soir	A	18:06	B	18:59
soir	A	19:26	B	20:19
nuit	A	20:46	B	21:31
jour	B	07:50	A	08:46
jour	B	09:10	A	10:06
jour	B	10:30	A	11:19
jour	B	12:30	A	13:19
jour	B	14:30	A	15:23
jour	B	15:50	A	16:43
soir	B	17:10	A	18:03
soir	B	18:30	A	19:23
soir	B	19:50	A	20:43
nuit	B	21:10	A	21:55
jour	A	07:22	C	08:30
jour	A	08:42	C	09:50
jour	A	10:02	C	11:01
jour	A	12:02	C	13:01
jour	A	14:02	C	15:05
jour	A	15:22	C	16:25
jour	A	16:42	C	17:45
soir	A	18:02	C	19:05
soir	A	19:22	C	20:25
nuit	A	20:42	C	21:36
jour	C	07:50	A	08:58
jour	C	09:10	A	10:18
jour	C	10:30	A	11:29
jour	C	12:30	A	13:29
jour	C	14:30	A	15:33
jour	C	15:50	A	16:53
soir	C	17:10	A	18:13
soir	C	18:30	A	19:33
soir	C	19:50	A	20:53
nuit	C	21:10	A	22:04

La vue *Instances* étends un *AbstractTableModel*. Elle présente les instances sous forme d'un tableau, et permet de les filtrer suivant leur lieu de départ, d'arrivée, ainsi que de leur service. Il est également possible de les ordonner.

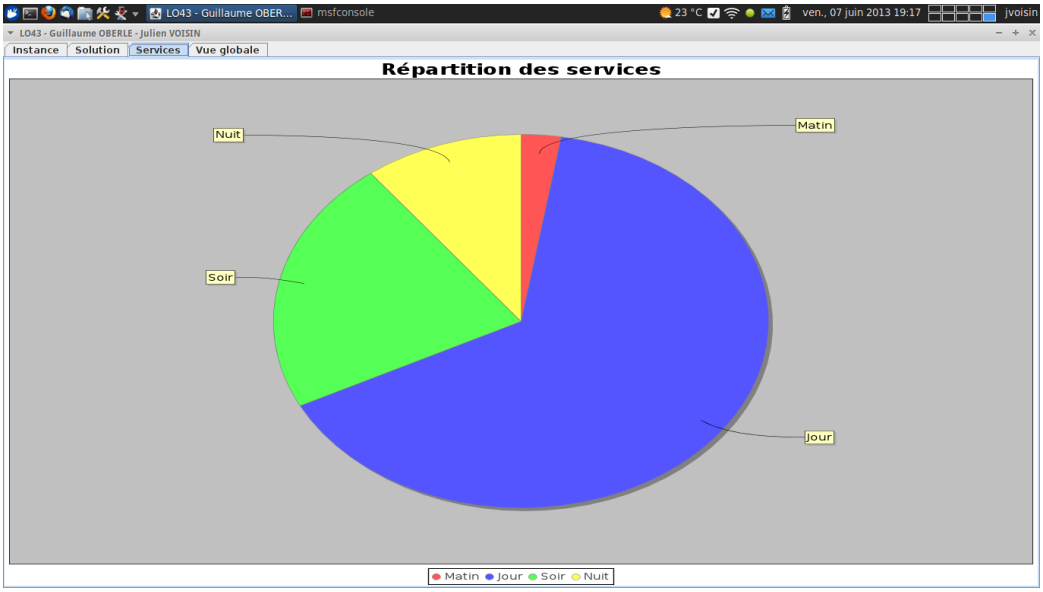
3.2.2 La vue *Solutions*



La vue *Solutions* permet d'afficher séparément les informations pour chaque chauffeur à l'aide d'une combobox. Elle présente le parcours du chauffeur sectionné sous forme d'un diagramme de Gantt, ainsi que dans un encart à droite trois types d'informations :

- Les informations sur le chauffeur, telles que son coût, son nombre de tâches, l'écart à la durée légale de travail, ses heures supplémentaires, ...
- Les informations sur la solution globale, telles que le nombre total de chauffeurs, son coût, et le nombre de tâches.
- Les informations légales, tirées du fichier de configuration, telles que la durée d'une pause, d'une journée, ...Ces informations permettent entre autre de calculer les dépassements et écarts à la loi possibles affichés dans les informations du chauffeurs.

3.2.3 La vue *Services*



La vue *Services* se présente sous forme d'un diagramme de type camembert, montrant la répartition en pour-cents des différents services (matin, jour, soir, nuit).

3.2.4 La vue *Globale*

La vue *Globale* montre sous forme de diagramme de Gantt la solution dans son ensemble.

4 Choix techniques

4.1 Parsage des données

Le parsage des données s'effectue à l'aide de la classe *Scanner*, et non pas avec des regexp. En effet, les fichiers étant formaté de manière stricte, il était plus facile d'utiliser la première méthode. De plus, les regexp en Java 1.6 ne sont pas vraiment confortables à utiliser.

4.2 La bibliothèque JFreeCharts

L'utilisation de cette bibliothèque a permis de réduire le temps de développement. Elle permet de créer des graphes de toutes formes (dont des diagrammes de Gantt) à partie de données brutes. Plusieurs raisons quant-à son utilisation :

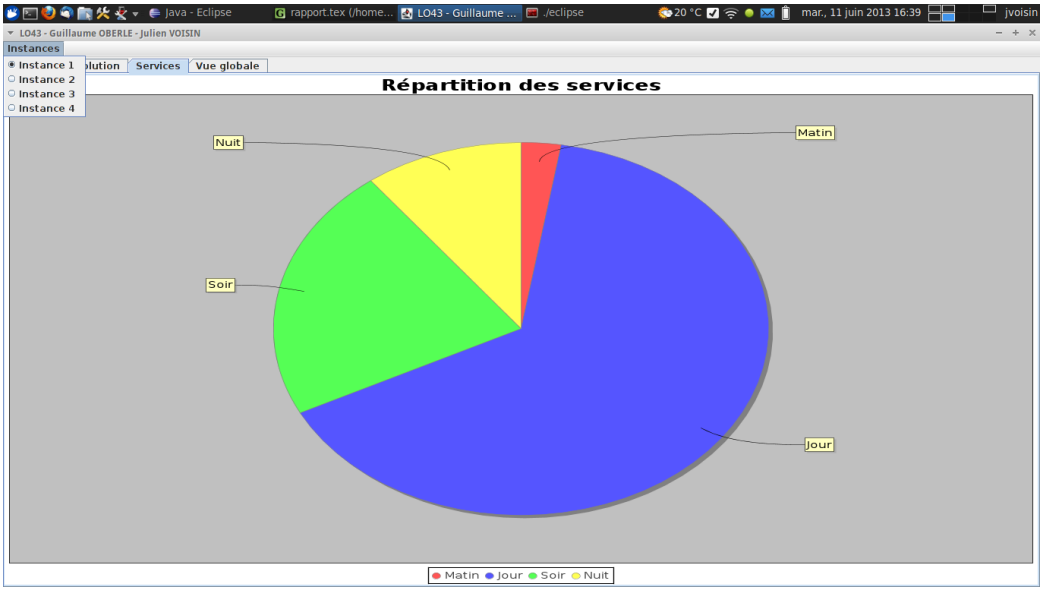
- Placée sous licence libre
- Maintenue
- Documentée
- Précédemment utilisée dans d'autres projets scolaires

4.3 Parallélisation

L'absence de parallélisation est un choix délibéré. En effet, le gain de vitesse et de confort utilisateur apporté est négligeable comparé à la complexité induite par son utilisation. Des tests ont même montré que l'utilisation de threads était parfois plus lente qu'un déroulement purement séquentiel.

4.4 Sélection des données

L'utilisation d'un menu sélecteur permet à l'utilisateur de choisir les données qu'il souhaite afficher. Le seul soucis de cette méthode est que le choix des données est codé en dur dans l'application. une amélioration possible serait de faire une boite de dialogue permettant à l'utilisateur de sélectionner ses jeux de données librement.



4.5 Outils de développement

Le développement de ce projet s'est fait à l'aide de l'éditeur *Eclipse*, et la gestion de version avec *git*, permettant de travailler de manière décentralisée sans crainte de conflits.

5 Conclusion

5.1 Problèmes rencontrés

Nous n'avons pas rencontré de problèmes majeurs lors du développement de ce projet. Deux soucis mineurs ont tout de même été à déplorer :

- Les fichiers de données fournies étaient mal formatés. Nous avons donc perdu du temps à essayer d'adapter notre parseurs aux différents cas particuliers.
- L'organisation de l'UV, notamment l'apprentissage du langage Java en fin de semestre ne nous a pas permis de le commencer tôt.

Le bilan de ce projet peut se concentrer sur deux points principaux : un bilan au niveau humain et au niveau pédagogique.

5.2 Bilans

5.2.1 Bilan humain

Au niveau humain, cela a permis de découvrir ou de redécouvrir l'esprit d'équipe dans un projet. Ce fut également l'occasion de s'apercevoir que pour peu que tout le monde maîtrise *git*, un projet peut se dérouler sans aucun problèmes.

5.2.2 Bilan pédagogique

Le bilan pédagogique est plutôt positif. Certains ont appris le Java, et l'ensemble du groupe en a profité pour améliorer ses connaissances de git ainsi que de LaTeX.

5.3 Conclusion finale

Le cahier des charge a été respecté, et le programme fonctionne correctement quel que soit le jeu de données en entrée. Un cahier des charges a été mené, la conception UML a été faite. Le développement a lui aussi été mené correctement puisqu'il s'est déroulé sans soucis. Il fut l'occasion de se perfectionner en java, ainsi qu'avec git.