Projet DevOO

Durée du projet : 5 séances sur 3 semaines

Encadrement : E. Egyed-Zsigmond, J. Salotti, C. Solnon

1 Description du système

L'application est inspirée d'un projet réel, piloté par le Grand Lyon entre 2012 et 2015, qui vise à optimiser la mobilité durable en ville (voir www.optimodlyon.com). Nous nous focaliserons ici sur la partie concernant le frêt urbain et l'optimisation des tournées de livraisons en ville.

Au lancement de l'application, l'utilisateur charge un fichier XML décrivant un plan de ville : ce plan donne les coordonnées des intersections et liste les tronçons de rues entre les intersections. Chaque tronçon a une longueur et une vitesse moyenne de circulation. Le plan chargé est visualisé à l'écran.

Quand un plan est chargé, l'utilisateur peut charger un fichier XML décrivant les demandes de livraisons à planifier. Chaque livraison a une adresse (correspondant à une intersection du plan) et une fenêtre de livraison. Une fenêtre de livraison est composée d'une heure de début et une heure de fin (postérieure à l'heure de début). Plusieurs livraisons peuvent avoir la même fenêtre de livraison. Si deux livraisons ont des fenêtres de livraison différentes, alors les fenêtres ne peuvent se chevaucher (autrement dit, l'heure de fin d'une des deux livraisons doit être inférieure à l'heure de début de l'autre).

Une fois que les demandes de livraison sont chargées, le système visualise la position de chaque demande de livraison sur le plan. L'employé peut alors demander au système de calculer une tournée. Cette tournée doit partir de l'entrepot (dont l'adresse est donnée dans le fichier décrivant le plan), passer par chaque demande de livraison, puis revenir à l'entrepôt. La durée d'une tournée est égale au temps nécessaire pour parcourir l'ensemble de ses tronçons, plus 10 minutes de temps d'arrêt pour chaque livraison (pour décharger les colis et les remettre au client). La durée de la tournée doit être minimale, tout en respectant les contraintes de précédence entre fenêtres de livraison : si deux livraisons l_1 et l_2 ont pour fenêtres de livraison $[d_1, f_1]$ et $[d_2, f_2]$ avec $d_1 < f_1 < d_2 < f_2$, alors l'itinéraire doit livrer l_1 avant de livrer l_2. Le système doit également calculer, pour chaque livraison, l'heure de passage de la tournée, en supposant que la tournée part de l'entrepôt à 8h00. L'heure de passage pour une livraison doit nécessairement être supérieure ou égale à l'heure de début de sa fenêtre de livraison (si le livreur arrive avant l'heure de début, alors il devra attendre l'heure de début pour effectuer la livraison). L'heure de passage pour une livraison peut être supérieure à l'heure de fin de sa fenêtre (s'il n'est pas possible d'effectuer toutes les livraisons dans les fenêtres prévues car il y a trop de demandes de livraison). Dans ce cas, le système signale les livraisons en retard à l'utilisateur.

La tournée calculée par le système est visualisée sur le plan (même si certaines livraisons ne peuvent être réalisées dans leurs fenêtres). Le système affiche également la liste des livraisons, dans l'ordre de la tournée, avec pour chaque livraison sa fenêtre de livraison et l'heure de passage prévue. L'utilisateur peut alors modifier interactivement la tournée (supprimer des livraisons, échanger l'ordre de deux livraisons, ou ajouter de nouvelles livraisons), et demander au système de modifier les horaires de passage en conséquence. Le système signale les livraisons pour lesquelles l'horaire de passage ne respecte pas la fenêtre horaire initialement demandée par le client. À tout moment, l'utilisateur peut demander l'annulation de modifications apportées à la tournée.

Lorsqu'une tournée a été validée, l'utilisateur peut demander la génération d'une feuille de route : une feuille de route est un fichier au format texte donnant la liste des livraisons à faire (dans l'ordre de passage de la tournée) et, pour chacune de ces livraisons, l'adresse de livraison, les heures prévues d'arrivée et de départ, et l'itinéraire à suivre pour rejoindre cette livraison depuis la livraison précédente ou depuis le dépôt.

2 Travail demandé

Le projet est réalisé en hexanôme, et un chef de projet sera chargé de la répartition du travail et de la coordination. Vous répartirez votre travail en fonction des besoins. Un planning prévisionnel indicatif vous est donné ci-dessous :

Séance 1 : Désignation du chef de projet ; élaboration d'un planning prévisionnel ; capture et analyse des besoins

Séance 2 : Conception

Séance 3 : Prise en main de l'environnement de développement ; implémentation et tests unitaires

Séance 4 : Implémentation et tests unitaires ; premiers tests d'intégration

Séance 5 : Fin de l'implémentation et des tests unitaires ; tests d'intégration ; génération de la documentation en ligne ; re-génération des diagrammes de classes et de packages à partir du code

Revue (semaine suivant la séance 5) : Démonstration du prototype (rendez-vous à fixer) et remise du compte-rendu.

Livrables :

- Capture et analyse des besoins
  - Planning prévisionnel du projet
  - Modèle du domaine
  - Glossaire
  - Diagramme de cas d'utilisation
  - Description textuelle structurée des cas d'utilisation
- Conception
  - Liste des événements utilisateur et diagramme Etats-transitions
  - Diagrammes de packages et de classes
  - Document expliquant les choix architecturaux et design patterns utilisés
  - Diagramme de séquence du calcul de la tournée à partir d'une demande de livraison
- Implémentation et tests
  - Code du prototype et des tests unitaires
  - Documentation JavaDoc du code
  - Diagrammes de packages et de classes retro-générés à partir du code
- Bilan
  - Planning effectif du projet
  - Bilan humain et technique

Environnement de développement : Nous vous proposons de développer en Java. Si vous souhaitez utiliser un autre langage orienté objet, vous devrez demander notre accord au préalable. Vous utiliserez un environnement de développement intégré (IDE) et des outils pour automatiser les tests unitaires, pour favoriser le travail collaboratif et la gestion des versions, et pour générer la documentation en ligne du code. Vous pourrez notamment utiliser l'IDE Eclipse [1] et ses plug-in ObjectAid [2], pour re-générer un diagramme de classes à partir du code Java, et JUnit [3], pour automatiser la réalisation des tests unitaires.

Vous pouvez utiliser ArgoUML [4] ou StarUML [5] pour saisir des diagrammes UML (diagrammes de classes, de packages, de séquences et états-transitions).

Vous générerez une documentation en ligne de votre code en utilisant JavaDoc, et vous suivrez le code de style préconisé par Oracle [6].

Pour calculer la tournée, vous pouvez utiliser le code Java disponible sur `liris.cnrs.fr/csolnon/TSP.jar`. La classe `TSP1` implémente la variante la plus basique des algorithmes vus en cours. Vous pouvez programmer des variantes plus élaborées en redéfinissant les méthodes `bound` et `iterator`.

1. `http://help.eclipse.org/juno/index.jsp` (voir notamment le Java Development User Guide)
2. `http://www.objectaid.com/`
3. `http://www.junit.org/`
4. `http://argouml.tigris.org/`
5. `http://staruml.io/`
6. `http://www.oracle.com/technetwork/java/codeconventions-150003.pdf`