PLD Agile

Durée du PLD : 8 séances de 4 heures

Equipe pédagogique :
— Groupes 1 et 2 : L. Groleaz, M. I. Popescu, M. Schiedermeier, C. Solnon
— Groupes 3 et 4 : E. Egyed-Zsigmond, M. I. Popescu, P.-E. Portier

1 Description du système

L'application est inspirée d'un projet réel, piloté par le Grand Lyon entre 2012 et 2015, et visant 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, et pour une mobilité encore plus durable, nous allons concevoir une application pour optimiser des tournées effectuées par des cyclistes.

Au lancement de l'application, l'utilisateur charge un fichier XML décrivant un plan de ville. Ce fichier donne la liste des intersections (avec, pour chaque intersection, sa latitude et sa longitude) et la liste des tronçons de rues (avec pour chaque tronçon, le nom de la rue, l'intersection de départ, l'intersection d'arrivée, et la longueur en mètres).

Quand un plan est chargé, le système le visualise et l'utilisateur peut alors charger un fichier XML décrivant une demande de livraisons. Ce fichier donne l'adresse de l'entrepôt et l'heure de départ de l'entrepôt. Il liste ensuite les livraisons à effectuer. Chaque livraison a une durée (en secondes) et une adresse. Les adresses (de l'entrepôt et des livraisons) correspondent à des intersections du plan.

Une fois qu'une demande de livraisons est chargée, le système visualise la position de chaque livraison sur le plan. L'employé peut alors demander au système de calculer un ensemble de tournées. Le nombre de tournées doit être égal au nombre de livreurs. Chaque tournée part de l'entrepôt et revient sur l'entrepôt après avoir visité des points de livraison. Toutes les tournées doivent comporter le même nombre de livraisons (à un près) et chaque point de livraison doit être visité par exactement une tournée. La durée d'une tournée est égale au temps nécessaire pour parcourir l'ensemble de ses tronçons, plus la durée de toutes ses livraisons. Pour calculer la durée nécessaire pour parcourir un tronçon, vous supposerez que tous les livreurs roulent à une vitesse constante de 15 kilomètres par heure. La somme des durées de toutes les tournées doit être minimale.

Les tournées calculées par le système sont visualisées sur le plan. Le système affiche également la liste des livraisons, avec pour chaque livraison les heures d'arrivée et de départ prévues. L'utilisateur peut alors modifier interactivement les tournées (supprimer des livraisons, ajouter de nouvelles livraisons, ou déplacer une livraison d'une tournée vers une autre). Après chaque modification, le système met à jour les horaires de passage des livreurs. À tout moment, l'utilisateur peut demander l'annulation de modifications apportées à la tournée.

2 Organisation

Le projet sera réalisé en hexanôme selon un processus de développement itératif. Le projet comportera au moins deux itérations, et la première itération durera 4 séances. Vous ferez une première démonstration de votre application à la fin de ces 4 séances. Lors des 4 séances suivantes, vous pourrez choisir de faire entre une et quatre itérations.

Première itération : Cette première itération correspond à la phase d'inception de la méthode UP. L'objectif est d'identifier les principaux cas d'utilisation, de concevoir une première architecture de votre application, et d'implémenter quelques cas d'utilisation afin d'avoir un premier noyau exécutable sur lequel le client pourra vous faire un retour.

Les livrables qui vous seront demandés à l'issue des quatre premières séances sont :
— Glossaire
— Diagramme de cas d'utilisation

— Description textuelle abrégée des cas d'utilisation
— Description textuelle structurée des cas d'utilisation qui ont été analysés et/ou implémentés
— Diagramme de packages et de classes
— Planning effectif de la première itération, détaillant pour chaque membre de l'équipe le temps passé sur chaque activité

Vous ferez une démonstration de votre prototype à la fin de la quatrième séance de projet.

Itérations suivantes : Au début de chaque itération suivante, vous choisirez les cas d'utilisation (ou les scénarios de cas d'utilisation) qui seront analysés, conçus et/ou implémentés pendant l'itération, en estimant pour chacun la durée prévue. A la fin de chaque itération, vous ferez une revue rapide pour récapituler ce qui a été réalisé en fonction des objectifs initiaux, et comparer les durées effectives avec les durées prévues.

Les livrables qui vous seront demandés à la fin du projet sont :
— Document expliquant les choix architecturaux et design patterns utilisés
— Code de votre application et des tests unitaires
— Documentation JavaDoc du code
— Diagrammes de packages et de classes rétro-générés à partir du code, accompagnés d'un commentaire sur les différences entre ces diagrammes et ceux élaborés lors de la première itération
— Rapport sur la couverture des tests
— Pour chaque itération : planning prévisionnel et planning effectif
— Bilan technique et humain du PLD

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, évaluer la couverture des tests, favoriser le travail collaboratif et la gestion des versions, et générer la documentation en ligne du code. Vous pourrez notamment utiliser l'IDE Eclipse [1] et ses plug-in ObjectAid [2] (pour rétro-générer un diagramme de classes à partir du code Java), JUnit [3] (pour automatiser la réalisation des tests unitaires), et EclEmma [4] (pour évaluer la couverture des tests).

Vous pouvez utiliser la version démo de 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 guide de style préconisé par Oracle [6].

Pour calculer une tournée, vous pouvez utiliser le code Java disponible sur Moodle. 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`, ou bien implémenter une autre approche de résolution (programmation dynamique ou recuit simulé, par exemple).

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