**Cahier des charges – Software Engineering**

**par Hugo STEIGER**

1) Contexte et définition du problème

Dans le cadre de ce projet, le client demande de réaliser un réseau générique qui pourra recouvrir la simulation des déclinaisons de réseaux suivantes :

- Réseau routier

- Réseau Web

- Réseau social

Voici une liste des prérequis demandés par le client quant à cette simulation :

- Interface graphique utilisateur intuitive

- Possibilité d’ajouter ou supprimer des objets via l’interface

- Possibilité de visualiser les croisements du réseau

- Possibilité de visualiser les éléments de priorité des flux du réseau

- Possibilité de visionner le réseau sous plusieurs échelles via l’interface

- La simulation doit prendre en compte le temps

- Gestion de flux aléatoires traités en temps réel

2) Objectif du projet

L’objectif du projet va être de réaliser un **application programmée de façon orientée objet** à l’aide de **Java** qui permettra de satisfaire les attentes du client listées précédemment. Il sera donc nécessaire de développer un ensemble de **quatre** **classes génériques** interagissant les unes avec les autres lors de la **simulation** et pouvant s’adapter à différents cas de figures. L’interaction de ces classes devra pouvoir être **observée et modulée** à l’aide d’une **interface graphique correspondant elle-même à une classe** et qui permettra de rendre compte de la simulation en respectant les contraintes explicitées.

3) Périmètre du projet

Un temps limité d’un mois est réservé à la réalisation de ce projet. Il est donc nécessaire de fixer un périmètre au projet pour remplir les contraintes du client sans se disperser.

À ce titre, le modèle du réseau choisi est relativement simple mais devrait permettre de vérifier la contrainte de **généricité** du client. En effet, un **réseau** peut se définir, de façon général, comme un ensemble de **nœuds** reliés entre eux par des **arcs** lestés d’un **facteur d’envoi**, limités par une **capacité d’envoi** et par lesquels transitent des **paquets** avec des **priorités** différentes, une **origine** et une **destination** propre ainsi qu’un **trajet** (ensemble de nœuds et d’arcs) permettant de relier les deux.

On peut démontrer que ce choix de définition à priori basique est pertinent et permet de bien délimiter le projet :

\* **Un réseau routier** peut être décomposé en différents points d’intérêts [nœuds], par exemple des entrepôts, qui sont reliés les uns aux autres par des routes [arcs]. Des colis [paquets] peuvent partir d’un entrepôt pour être acheminés vers un autre [origine / destination], et ce avec un certain choix de trajet et potentiellement un certain ordre de priorité (on donnera l’exemple des lettres prioritaires et des lettres classiques qui n’ont pas la même priorité d’envoi). Selon la longueur et le type de la route, le temps pris est plus ou moins important [facteurs d’envoi] et une petite route peut facilement être saturée par un bouchon si trop utilisée alors qu’une grosse autoroute n’aurait pas ce problème [capacité d’envoi].

\* **Un réseau Web** peut aussi être décomposé en nœuds (comme des serveurs ou des clients) qui sont reliés entre eux par Internet, où les données [paquets] empruntent des chemins bien particuliers [arcs] pour transiter (e.g. données GPS via satellite). Selon la route choisie [trajet] par un paquet de donnée, celui-ci peut arriver, selon les arcs utilisés, plus ou moins rapidement [facteurs d’envoi] d’un client vers un serveur e.g. [origine / destination]. On peut choisir la priorité des données à envoyer. Passer par un décodeur avec une forte bande passante permettra à l’utilisateur uploader plus de donnée à la seconde qu’une route utilisant la 3G d’un téléphone e.g. [capacités d’envoi].

\* **Un réseau social** est composé de personnes [nœuds]. Ces personnes peuvent communiquer à des personnes cibles [origine / destination] par l’intermédiaire d’autres personnes [trajet] car elles se connaissent [arcs]. La communication est constituée de message [paquets] qui peuvent passer d’une personne à une autre. Cette communication est limitée par l’affinité des personnes en contact [facteur d’envoi] : des personnes se faisant la guerre auront plus de difficultés à communiquer entre elles (hors insultes). Certaines personnes sont plus ouvertes à la discussion que d’autres plus concises [capacité d’envoi]. Enfin, des messages urgents concernant un accident devront être transmis plus rapidement qu’une blague [priorité d’envoi].

La **simulation du réseau**, sous ses différentes formes, utiliseraient donc ces notions simples et génériques qui, en interaction, pourront mener à la mise en place de systèmes plus complexes mais suffisamment encadrés. Le temps peut d’ailleurs être correctement pris en compte avec les notions de facteurs, capacités et priorités d’envoi.

Au niveau de **l’interface graphique** interactive et de la simulation y apparaissant en temps réel, le peu d’objets introduits limite la complexité du système et permet d’avoir un rendu plus fluide. L’interface graphique fonctionnera ainsi :

- Elle sera composée d’une fenêtre décomposé en 3 partie : la fenêtre de visualisation, une palette d’outil et une barre de navigation.

- La barre de navigation contient un bouton pour changer le type du réseau visualisé, un bouton pour reset et un bouton pour lancer un flux aléatoire.

- La palette d’outil permet d’activer certaines options. L’utilisateur pourra alors interagir d’une certaine manière avec la fenêtre de visualisation avec sa souris. La palette d’outil contient : ajouter nœud, détruire nœud, ajouter arc, détruire arc, ajouter paquet, détruire paquet, détails, navigation. Avec la plupart de ces options, clic gauche sur l’objet à détruire ou à l’endroit où on souhaite le créer fait l’action indiquée. Pour une création de nœud, on clique sur un endroit du plan, pour la destruction on clique sur le nœud en question. Pour une création d’arc, on sélectionne deux nœuds, pour la destruction, on clique sur l’arc. Pour la création d’un paquet, on sélectionne un nœud d’origine. Pour sa destruction, on sélectionne le paquet à détruire. Lors de sa création, un clic sur un autre nœud permet de donner une destination au paquet et la route est choisie avec un plus court chemin. Cependant celle-ci peut aussi être générée avec le bouton flux aléatoire, qui ne suit pas nécessairement le plus court chemin mais un chemin quelconque vers une destination aléatoire. On peut naviguer avec l’option navigation en se déplaçant avec des clics gauches et en zoomant / dézoomant avec la molette de la souris. L’option détails permet d’avoir avec un clic gauche sur un objet des informations le concernant.

- La fenêtre de navigation décrit le réseau en temps réel et l’affichage varie selon le type de réseau choisi. Tous les types d’objets y sont représentés et peuvent être étudiés / modifiés avec la palette d’outil.

4) Description fonctionnelle des besoins

On utilisera les **classes, attributs et méthodes** suivantes pour permettre la simulation des différents réseaux.

**Classe Nœud :**

Attributs :

* Règles de priorité
* Liste fil de priorité

# Pour faire le choix de quel paquet en entrée d’un nœud envoyer en premier à sa sortie et sur quel arc, on utilise une fil de priorité avec des règles spécifiques au niveau du nœud qu’il faut préciser

* Identité du nœud

Méthodes :

* Obtention des nœuds voisins

# Les nœuds voisins d’un nœuds A sont ceux connectés par au moins un arc avec A

* Ajouter un paquet dans la liste de priorité
* Retirer un paquet de la liste de priorité
* Obtenir attributs

**Classe Arc :**

# Les arcs seront supposés orientés, ainsi A->B != B->A

Attributs :

* Facteur d’envoi
* Capacité d’envoi
* Nœud de départ
* Nœud d’arrivée

Méthodes :

* Mise à jour facteur d’envoi
* Mise à jour capacité d’envoi
* Obtenir attributs

**Classe Paquet :**

Attributs :

* Nœud position
* Nœud destination
* Nœud de départ
* Priorité
* Trajet

Méthodes :

* Mise à jour priorité
* Obtenir attributs

**Classe Réseau :**

# Cette classe utilise toute les autres pour créer en ensemble cohérent et en interaction

Attributs :

* Liste des instances nœuds
* Liste des instances arcs
* Matrice d’adjacence

# La matrice d’adjacence simplifie la représentation et la modification des liens entre nœuds et arcs dans le réseau

* Liste des instances paquets
* Temps
* Type de réseau

Méthodes :

* Plus court chemin
* Ajouter Nœuds
* Ajouter Arcs
* Ajouter Paquet
* Obtenir attributs

**Classe Interface :**

Attributs :

* Temps
* Réseau

# Le réseau est généré aléatoirement à chaque lancement de l’interface, aucun paquet ne s’y trouve cependant

* Type de réseau

# Permet d’interpréter le réseau comme étant routier, Web ou social. Ne se fait qu’au niveau de l’interface

* Position du centre
* Échelle
* Option palette activée

# Par défaut, l’option activée est la navigation

Méthodes :

* Reset
* Modifier le type de réseau
* Lancer un flux aléatoire
* Zoomer
* Dézoomer
* Ajouter un nœud
* Détruire un nœud
* Ajouter un arc
* Détruire un arc
* Ajouter un paquet
* Lancer un flux déterministe
* Détruire un paquet
* Détails nœud
* Détails arc
* Détails paquet

\* Représentation par diagramme UML (à venir)

5) Répartition du travail en tâches

\*Travail administratif

Responsable du projet : Cécile Aprili

Rédacteur : Hugo Steiger

\*Groupes de travail

Classe Nœud : Célian Muller-Machi, Aymane Moataz, (Romain Bouhier)

Classe Arc : Jean Sikora, Nathan Mandin, (Paul Aldeano)

Classe Paquet : Jaafar Kabouche, Gaspard Houillon, (Bastien Eglem)

Classe Réseau : Sylvain Geiser, Cécile Aprili, (Hugo Steiger)

Classe Interface : Hugo Steiger (Implémentation de la base graphique de l’interface et de l’affichage de base du réseau), Romain Bouhier (Implémentation des nœuds), Bastien Eglem (Implémentation des paquets), Paul Aldeano (Implémentation des arcs)

**Le travail pour la classe Interface sera traité après qu’une base pour les classes implémentées ait été réalisée par les groupes s’en occupant**. Ainsi, au départ, le groupe classe Interface est dispatché dans les autres groupes correspondant à l’implémentation graphique qu’ils auront à faire. La base graphique étant liée à la vision globale du réseau, la personne qui s’occupera de son développement travaillera au départ avec la classe Réseau, mais pourra commencer par l’aspect graphique.

Les méthodes des classes, s’ils elles doivent être utilisées entre groupe, seront considérées comme des **boîtes noires** (une entrée, on se convainc que c’est la bonne sortie jusqu’à ce que la vraie méthode soit implémentée).

En plus de son travail de développement, **le rôle de la responsable de projet consistera à vérifier que cette cohérence entre les groupes est atteinte** pour avancer de façon fluide.

\*Répartition globale du travail au sein de chaque groupe (à venir)

6) Délais et planning prévisionnel

Ce planning se fonde sur **la méthode Agile**.

Séance 4 - 4 novembre : Création du groupe de travail, réflexion de groupe sur la problématique, sélection d’une responsable de groupe et d’un rédacteur, réalisation d’un premier cahier des charges peu détaillé

Entre 4 et 5 : Cahier des charges détaillé avec répartition des tâches et explication des choix faits.

Séance 5 - 25 novembre : Courte réunion de démarrage pour récapituler les rôles de chacun, le contenu des classes à réaliser et l’interaction entre les groupes de travail (notions de boîte noire). Répartition du travail dans chaque groupe et début du travail sur les classes.

Entre 5 et 6 : Répartition des tâches pour chaque personne dans chaque groupe détaillée dans le livrable. Mise à jour du livrable après les réflexions de la semaine dernière quant à chaque tâche à réaliser + diagrammes UML à finaliser.

Séance 6 - 2 décembre : Bilan de l’avancement lors de la réunion flash (notamment les problématiques rencontrées et les modifications à apporter à certains concepts). Continuité du travail, début de l’implémentation de l’interface graphique à partir des classes.

Entre 6 et 7 : Précisions des éléments réalisés jusqu’à maintenant dans le livrable.

Séance 7 - 9 décembre : Bilan de l’avancement lors de la réunion flash. Build quasiment terminé à obtenir à la fin de la séance (quasi fonctionnel pour au moins un type de réseau).

Entre 7 et Soutenance – 9 janvier : Build final à compiler (sans bug, avec plus de types de réseau si possible), Rapport de projet à réaliser ainsi que le support de soutenance.