Institut **S**upérieur d'**I**nformatique de
Modélisation et de leurs **A**pplications

Campus des Cézeaux
BP 10 125
63 173 AUBIERE CEDEX

Rapport d'ingénieur
projet de 2ème année
Génie logiciel et Systèmes Informatiques

Modélisation de modèles métiers orientés objets développés pour les réseaux ferrés français

(TOME I)

Présenté par: **Maxime ESCOURBIAC et Jean-Christophe SEPTIER**

Responsable ISIMA : David HILL

date de la soutenance : Mercredi 23 mars

Remerciements :

Nous tenons à remercier notre responsable de projet, David Hill, qui nous a prodigué des conseils et nous a fait part de son expérience tout au long de ce projet.

Table des figures et illustrations

Résumé

Le sujet du projet était d'étudier un modèle proposé par le RFF (Réseaux Ferrés de France). Pour réaliser ceci, nous avons dû étudier le logiciel de modélisation utilisé ici par l'entreprise, ArgoUML et faire un tutorial montrant les bases de son utilisation.

Pour compléter cette étude, nous avons effectué un travail de recherche sur des publications, réalisées durant ces dernières années, par des chercheurs de nombreux pays.

Cette étude préliminaire nous a permis d'implémenter un simulateur de transports ferroviaires en C# sous Visual Studio 2010 utilisant une simplification du modèle proposé par le RFF.

The topic of the project is studying a model proposed by RFF ((Réseaux Ferrés de France). For realize that. We had to study the software used by RFF, ArgoUML, and to do a tutoriel showing how to use it.

In complement, we did a research on scientific publication realized during the last years by scientists of many countries

This preliminary study, permitt us to implement un railway simulator which use a simplification of the model given by RFF

Table des matières

Introduction

Dans un soucis d'améliorer ses services et l'optimisation de son réseau pour ses clients, réseau ferré de France (RFF) a décidé de lancer une demande d'offre à certains laboratoires afin d'améliorer son modèle de données. Dans ce contexte, il nous a été demandé d'étudier le modèle existant et l'environnement actuel de la recherche dans le domaine des transports.

Dans le cadre de notre projet de deuxième année, nous avons, dans un premier temps, dû nous former à un outil de modélisation, ArgoUml, afin d'en connaître les avantages et les inconvénients, et de savoir si cet outil peut être utilisé pour cette étude. Par la suite nous avons effectué une recherche sur le contexte et sur les différentes études réalisées dans le monde. L'objectif de cette partie était de connaître ce qui existe déjà dans le domaine de la recherche sur les problèmes de transport, mais aussi les normes et les contraintes qui peuvent exister.

Une fois cette étude préliminaire finie, le modèle RFF nous a été fourni. Une compréhension du mécanisme et du vocabulaire spécifique du domaine du transport ferroviaire était nécessaire afin de pouvoir faire une critique de la proposition initiale et ainsi d'en tirer les avantages et les inconvénients.

Pour illustrer cette étude, il nous a été finalement demandé de réaliser une simulation, dont le cahier des charges ne nous a pas été imposé. Elle nous permettra ainsi d'appuyer nos remarques précédentes même si le modèle utilisé sera simplifié.

Dans ce projet, nous nous sommes concentrés à répondre aux questions suivantes. Est-ce-que cet outil est-il adapté pour ce type d'utilisation ? Comment fonctionne le modèle actuel et comment pourrait-il être amélioré ? Dans quel contexte international se situe cette étude

Nous commencerons donc par expliquer comment utiliser ArgoUml et quels sont ses défauts et ses avantages. Nous expliquerons ensuite quels sont les normes et les modèles existants. Nous décrirons le modèle de RFF, avant de présenter la simulation que nous avons réalisé.

I Etude d'ArgoUml

I.1 Présentation générale:

ArgoUML est un logiciel de création de diagrammes UML, programmé en Java. Il est sous licence Eclipse Public Licence (EPL) 1.0. Il permet la création de 7 types de diagrammes différents:

- Diagramme de cas d'utilisation.
- Diagramme de classe.
- Diagramme de séquence.
- Diagramme de collaboration.
- Diagramme d'état.
- Diagramme d'activité.
- Diagramme de déploiement.

Parmi les fonctionnalités existantes, il existe la génération automatique de code des classes définies dans le diagramme de classe. Cela peut se faire en plusieurs langages: C++, C#, Java, PHP5. Le code généré contient aussi des éléments de documentation compatible avec Doxygen.

I.2 Installation

Dans cette partie, nous allons expliquer comment installer argoUML sur les 3 systèmes les plus répandus: Windows,Linux et Mac.
Tous les fichiers sont disponibles à l'adresse URL suivante :
http://argouml-downloads.tigris.org/argouml-0.30.2/

1-Windows:

Un installateur .exe est disponible, donc l'installation est simple et se fait comme n'importe quel programme windows.

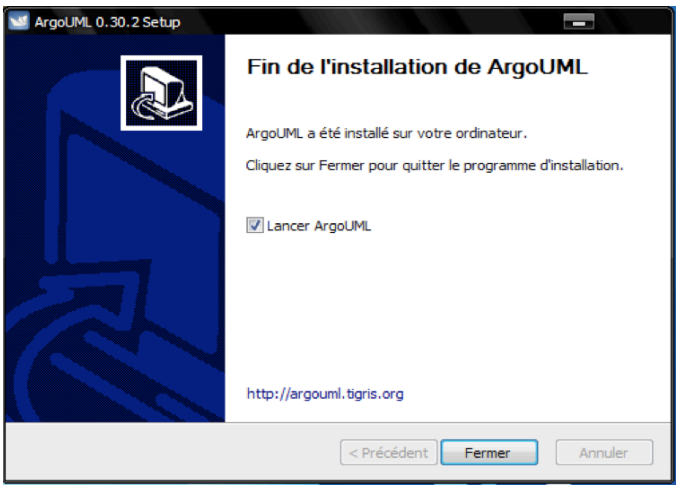


fig 1 : installation argoUML windows

2-Linux:

Pour ce système, il existe une version binaire, donc pour l'installer, il faut effectuer les commandes suivantes.

On décompresse l'archive téléchargée:

```
escourbi@escourbi-desktop:~/Bureau$ tar -xvf ArgoUML-0.30.2.tar.gz
```

fig 2 : installation argoUML Linux partie 1

On copie le répertoire décompressé dans /usr/bin/ :

```
escourbi@escourbi-desktop:~/Bureau$ sudo mv ./argouml-0.30.2/ /usr/bin/
```

fig 3 : installation argoUML Linux partie 2

On peut maintenant lancer argoUML avec la commande suivante:

```
escourbi@escourbi-desktop:~/Bureau$ /usr/bin/argouml-0.30.2/argouml.sh
```

fig 4 : installation argoUML Linux partie 3

3-Mac:

L'installation d'argoUML se fait comme toutes autres applications MAC, il suffit d'effectuer un Drag&Drop[1] dans le dossier 'Applications'.
Le fichier d'archive en .tar n'est pas nécessaire au fonctionnement de l'application. Il contient toutes les documentations et les aides existantes.

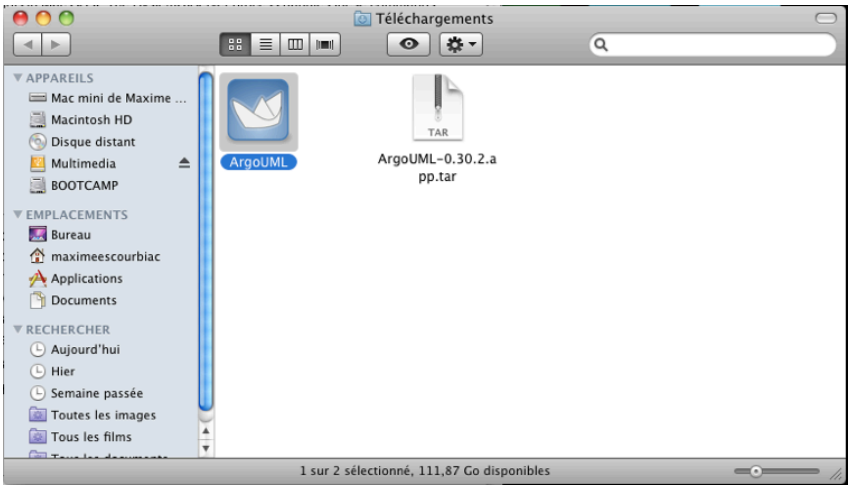


fig 5 : installation argoUML Mac

[1] Glisser - déposer

I.3 Utilisation

Dans cette partie, on verra comment aborder, et ensuite utiliser argoUML pour trois diagrammes.

1-Présentation générale de l'interface:

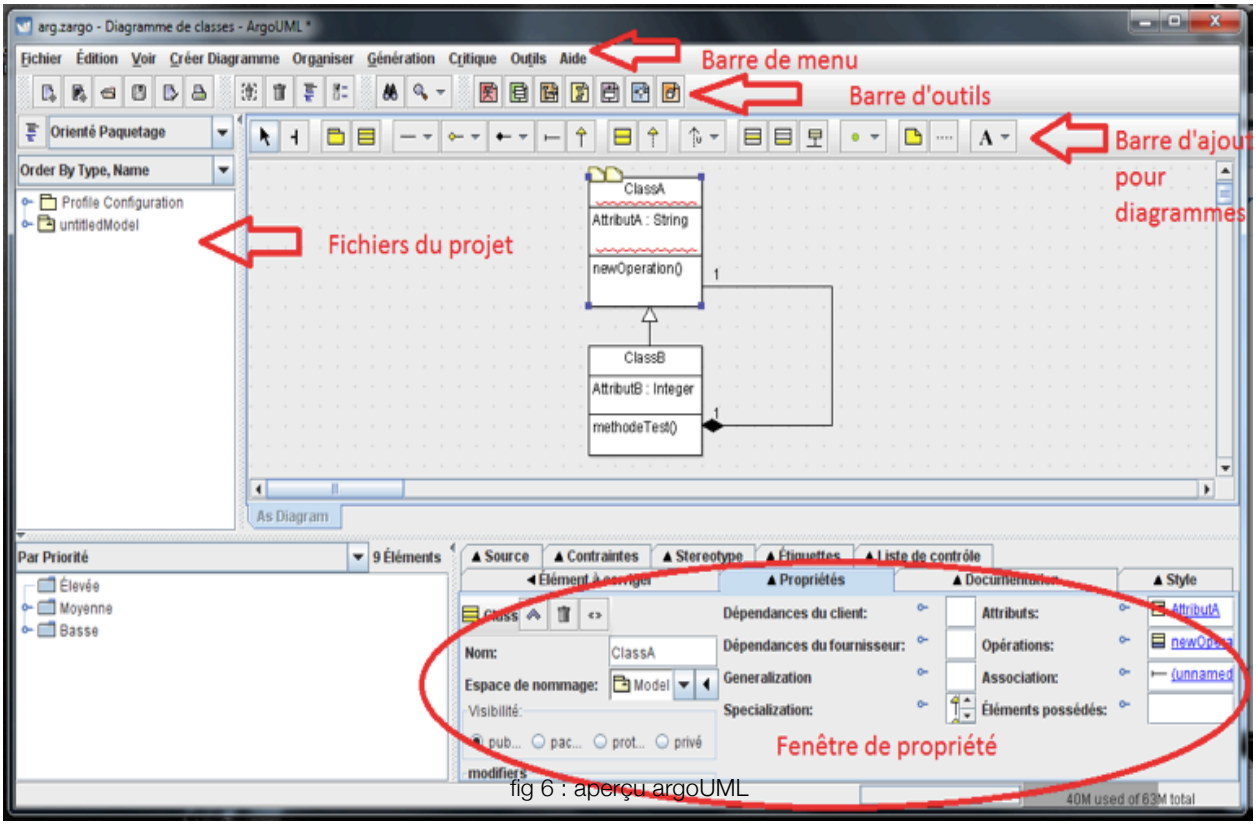


fig 6 : aperçu argoUML

fig 6 : interface principale d'argoUML

Nous voyons sur la capture d'écran (ci-dessus) les différentes parties d'argoUML.

-La barre de menu : elle permet de gérer les options d'argoUML.
-La barre d'outils: elle permet de créer, ouvrir ou enregistrer le projet, ainsi que d'autres outils. Les icônes à droite permettent de passer d'un type de diagramme à un autre.
-La barre d'ajout pour les diagrammes: elle permet de créer le diagramme, en proposant les différents symboles d'un diagramme.
-La fenêtre de hiérarchies des fichiers du projet, qui permet de retrouver les différents symboles d'un diagramme.
-La fenêtre de propriétés permet l'affichage des propriétés des éléments sélectionnés.

2-Création d'un diagramme de classe:

2.1-Ajout des éléments au diagramme :

Pour créer un diagramme de classe UML, on créé un nouveau projet puis on choisit le diagramme voulu (ici le diagramme de classe). On obtient alors l'ensemble d'icônes suivant.

fig 7 barre d'ajout pour le diagramme de classe

La première icône permet de passer en mode sélection.
La deuxième permet de déplacer tous les éléments en même temps. On peut l'utiliser pour remettre en forme rapidement le diagramme.
La troisième permet de créer un package.
La quatrième crée une nouvelle classe.
Pour ajouter un attribut ou une méthode à la classe, on double-clique sur la classe correspondante. On ajoute ensuite le nom puis le type pour les attributs et le prototype pour les méthodes dans la fenêtre de paramètres. Il est possible ainsi de choisir sa visibilité.
Les 3 icônes suivantes permettent de créer des relations, des agrégations ou des compositions. Pour créer un de ces éléments, il suffit de relier deux classes entre elles. (On verra comment les réaliser ci-dessous.)
La neuvième permet de spécifier une spécialisation d'une classe.
La dixième permet de créer des interfaces.
La seizième permet la création de nouveau type de données.
Et les commentaires sont gérés par les 3 dernières icônes.
Pour relier les classes entre elles, il suffit de cliquer sur l'icône de la relation choisie, et de relier les deux classes entre elles. On peut donc ajouter une relation, une composition, une agrégation ou encore une spécification.

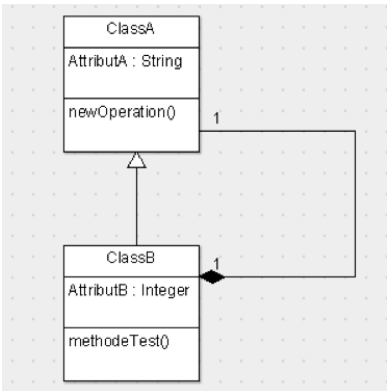


fig 8 : Création d'association entre les classes

On peut aussi créer une association ternaire, il faut créer d'abord une relation, puis relier la 3ème classe avec la relation. Un losange se forme. On peut rajouter ainsi d'autres classes.

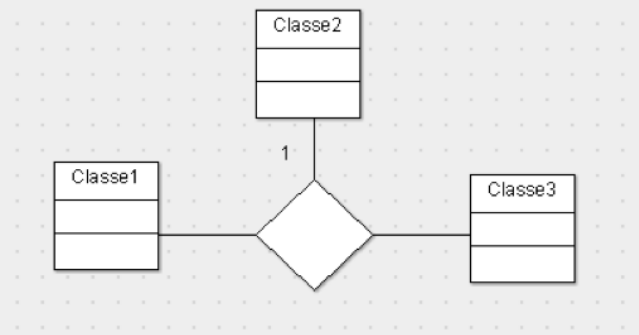


fig 9 : Les associations ternaires

2.1-Propriétés:

Chaque type d'élément (classe, association, etc...) possède ses attributs propres. Ils peuvent être modifiés en cliquant sur l'élément voulu. Les différentes propriétés s'affichent alors dans la fenêtre en dessous. Par exemple, pour une classe, on peut choisir la visibilité, les attributs, les opérations et le nom de la classe.

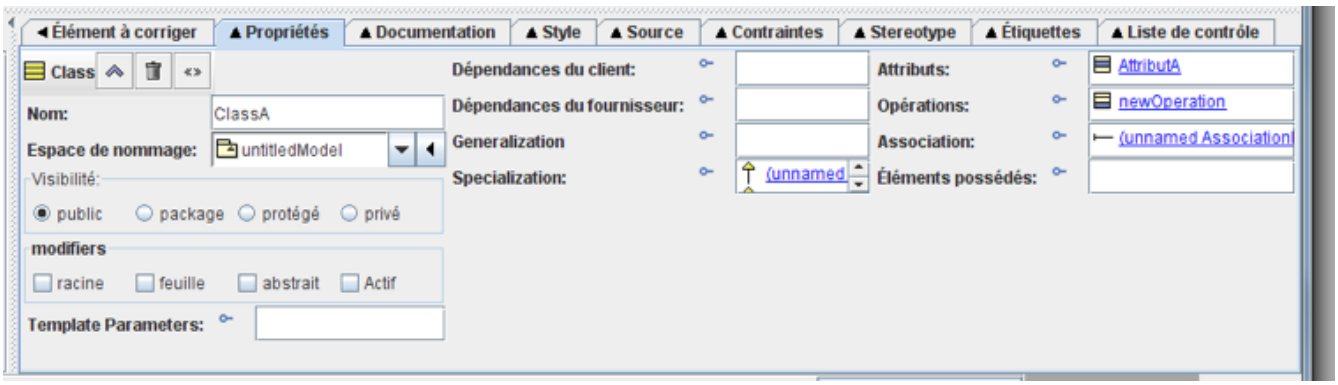


fig 10 : onglet propriétés d'argoUML

La documentation peut être ajoutée/modifiée dans l'onglet « Documentation » de façon analogue à l'onglet des propriétés.
La génération de code se fait automatiquement. Il faut cliquer sur « générer » (dans la barre de menu, puis « générer toutes les classes ». On choisit ensuite le langage dans lequel on veut générer le code, puis on clique sur générer. Le fichier de code sont alors créer dans le dossier correspondant. On peut ensuite lancer Doxygen pour créer la documentation.

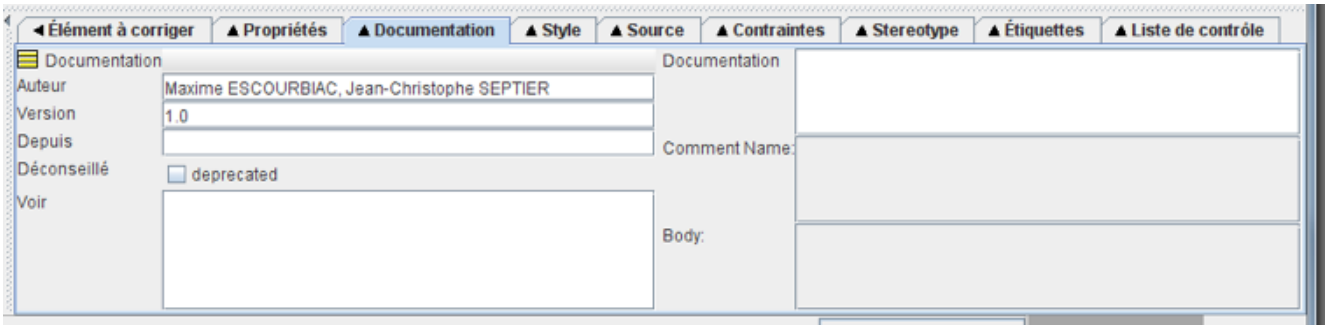


fig 11 : onglet documentation d'argoUML

Par exemple, pour l'exemple précédent, le code généré pour la classe A est le suivant :

ClasseA.hpp

```
#ifndef ClassA_h
#define ClassA_h

class ClassB;

    /** @author Maxime ESCOURBIAC, Jean-Christophe SEPTIER
     *  @version 1.0
     */
class ClassA {
 public:
    /**
     *  Methode de test de la classe A
     */
    virtual void newOperation();

 public:

    /**
     *  String attribut de la classe A
     */
    String AttributA;

 public:

/** @author Maxime ESCOURBIAC, Jean-Christophe SEPTIER
     *  @version 1.0
     */
    ClassB *myClassB;
};

#endif // ClassA_h
```

ClasseA.cpp

```
#include "ClassA.h"


    /** @author Maxime ESCOURBIAC, Jean-Christophe SEPTIER
     *  @version 1.0
     */

    /**
     *  Methode de test de la classe A
     */
void ClassA::newOperation()
{
```

fig 12 : La documentation sous argoUML

3-Création d'un diagramme de cas d'utilisation:

Comme pour le diagramme de classe tout se fait sur la barre d'ajout. Pour le diagramme de cas d'utilisation elle se présente de la façon suivante.

fig 13 : Barre d'ajout pour le diagramme de cas d'utilisation

La première icône permet de passer en mode sélection.
La deuxième permet de déplacer tous les éléments en même temps. On peut l'utiliser pour remettre en forme rapidement le diagramme.
La troisième permet d'insérer un acteur dans le diagramme.
La quatrième permet d'insérer un nouveau cas d'utilisation.
La cinquième est un menu déroulant pour le choix d'association entre les acteurs et les cas d'utilisation.
La sixième permet de gérer les dépendances.
La septième permet de créer des spécialisations des acteurs ou de cas d'utilisation.
La huitième et le neuvième permettent de gérer les inclusions et les extensions des cas d'utilisation.
Les dernières icônes servent à la documentation du diagramme.

Les principes pour relier des acteurs et des cas d'utilisation et pour afficher/modifier les propriétés sont équivalents au diagramme de classe.

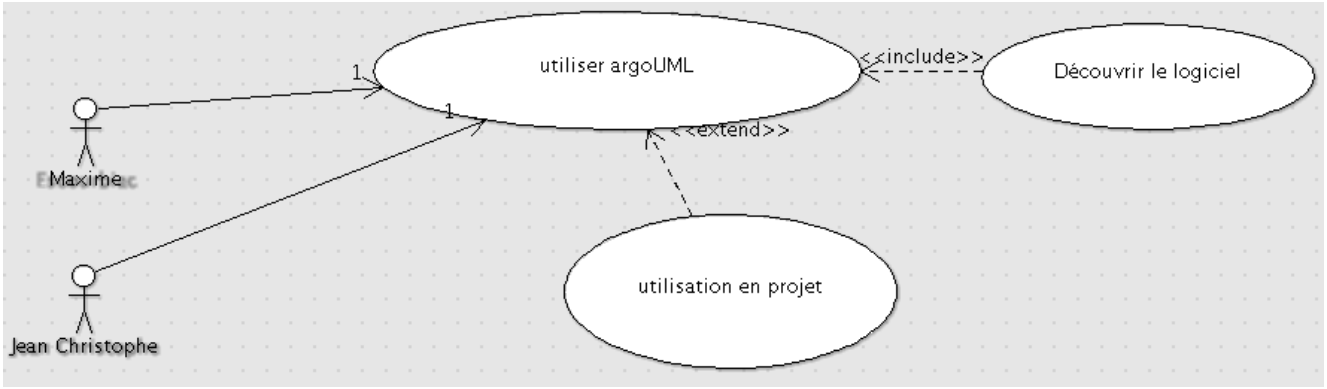


fig 14 : Exemple de diagramme de cas d'utilisation

4-Création d'un diagramme d'état:

Comme pour le diagramme de classe et de cas d'utilisation tout se fait sur la barre d'ajout. Pour le diagramme d'état, elle se présente de la façon suivante.

fig 15 : Barre d'ajout du diagramme d'état

Le premier sert à créer un nouvel état.
Le second sert à créer un nouvel état composite.
Le troisième sert à créer une nouvelle transition entre deux états.
Le quatrième sert à synchroniser plusieurs états.
Le cinquième et le sixième permettent de créer des sous états et des conteneurs d'états.
Le septième et le huitième insèrent les états initiaux et finaux du diagramme d'état.
Le neuvième et le dixième servent à gérer les jonctions et les choix.
Le onzième et le douzième permettent les états parallèles (fork).
Le treizième et le quatorzième sont pour la gestion des historiques.
Les derniers annotés sont pour la gestion des événements et des actions.

Les principes pour relier les états entre eux ou avec des opérateurs, et pour afficher/modifier les propriétés sont équivalents au diagramme de classe et au diagramme de cas d'utilisation.

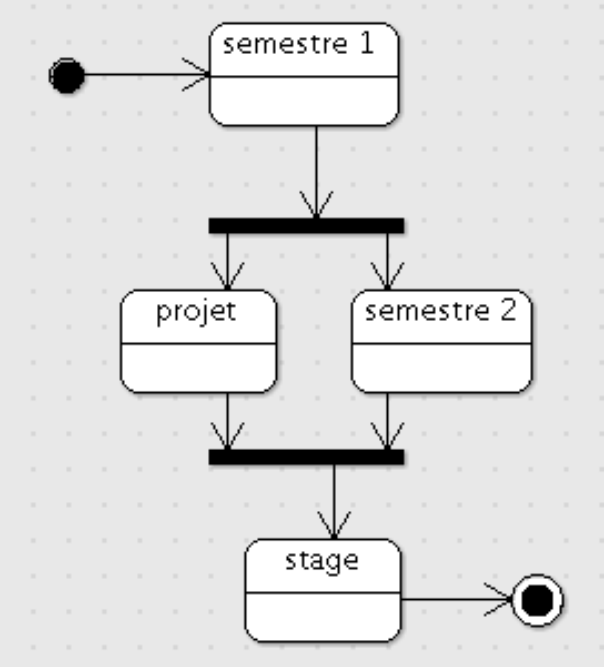


fig 16 : Exemple de diagramme d'état

5-Analyse des points positifs et négatifs:

Le premier point positif est la licence Eclipse Public Licence (EPL) qui permet d'utiliser gratuitement un logiciel de création de diagramme UML. On peut rappeler que la licence EPL n'est pas dite «virale», c'est à dire que l'on peut l'utiliser sans être obligé d'ouvrir les modèles sur lesquels on travaille.
Le deuxième est la grande facilité d'utilisation. En effet, la création des différents diagrammes est intuitive, et les interfaces des divers éléments se ressemblent.

On peut ajouter la génération de codes en plusieurs langages. Avec les mêmes diagrammes de classe UML, il est possible de créer les fichiers de codes commentés dans différents langages. La documentation du code permet d'utiliser directement Doxygen.

Dans les points négatifs, on peut noter l'austérité de l'interface, qui n'est pas très moderne. On peut aussi regretter l'absence de raccourcis clavier standards à la plupart des programmes (Ctrl+c, Ctrl+v, Ctrl+z etc...) qui ralentit considérablement la conception du modèle car tout doit se faire à la souris.

Un autre défaut notable présent dans les diagrammes de classe est l'absence d'un indicateur sur la visibilité des classes (les symboles +,-, # utilisés en UML2) qui peut gêner sur les diagrammes de classe de conceptions (méthodes et attributs).

On pourrait rajouter l'absence de gestion de travaux collaboratifs, qui aurait pu se faire avec un module pour travailler avec des gestionnaires de projet comme SVN, GIT. Un support de ce type de technologie pourrait permettre une adoption plus grande de ce logiciel dans le monde de l'entreprise.

Cependant ces défauts n'altèrent en rien la qualité générale du programme et ce logiciel peut servir à concevoir des modèles d'une grande complexité.

II Etude des modèles et des normes existantes

II.1 : Les normes existantes.

Dans le domaine du transport, comme dans de nombreux domaines industriels et scientifiques, il existe des organismes qui ont pour but de fixer des normes afin de permettre une interopérabilité entre les différents systèmes.
On a une organisation hiérarchique basée comme ceci:

1-Un organisme mondial, l'ISO (International Organization for Standardization):

Fondé le 23 février 1947, l'ISO est une organisation non-gouvernementale présente dans 163 pays, qui a pour but de produire des normes internationales dans les domaines industriels et commerciaux appelées normes ISO**.**
(Source: www.iso.org)

fig 17 : Logo de l'ISO

Dans le domaine des transports publics, il existe 16 groupes de travail, qui ont tous un rôle bien défini, allant de la partie technique à la partie commerciale. En voici la liste:

Numéro	Description du groupe de travail	Localisation
1	Architecture	Royaume-Uni
2	Exigence qualité et sécurité	Japon
3	Technologie de base de données	Japon
4	Identification automatique des véhicules	Norvège
5	Prélèvement des charges et des prix du billet	Pays-Bas
7	Gestion du fret et de la flotte commerciale	Canada
8	Secours et transport public	États-Unis
9	Contrôle et management du système d'information	Australie
10	Système d'informations pour les voyageurs	Royaume-Uni
11	Système de navigation et de guidage	Allemagne
14	Alarme véhicule/route et systèmes de contrôle	Japon
15	Communications courtes portées pour les applications TICS	Allemagne
16	Communications longues portées/ Protocoles et interfaces	États-Unis

fig: 18 Les différents groupes de travail de l'ISO

Pour la suite on va parler des organismes qui nous concernent directement, c'est à dire les organismes européens et français. Il existe un organisme normalisateur dans chaque continent et dans la plupart des pays développés.

2-Un organisme européen, le CEN (European Committee for Standardisation):

Cet organisme possède le même but que l'ISO, mais à échelle continentale. Le CEN a été fondé en 1961, il comprend actuellement 31 pays membres. Le CEN s'est associé à l'ISO en 1991 avec le traité de Vienne.

fig 19: Logo du CEN

Le CEN oeuvre aussi en groupe de travail décomposé comme ci-dessous:

Numéro	Description du groupe de travail
1	Contrôle des accès et paiement automatique
2	Système de management de la flotte et du fret
3	Transport Public
4	Information sur le trafic et le voyageur
5	Contrôle du trafic
6	Gestion du parking (stockage de la flotte)
7/8	Base de données des voies et sur leurs statistiques
9	Communications courtes portées
10	Interface homme-machine
11	Interfaces sous-système et inter-système
12	Identification automatique des véhicules et des équipements
13	Architecture système et terminologie

fig 20 : Les groupes de travail du CEN

3-Un organisme national, l'AFNOR (Organisation Française de Normalisation):

L'AFNOR est la SDO (Standards Development Organisations) française. Elle est composée de plusieurs groupes spécialisés dans un domaine particulier. L'AFNOR permet de coordonner et de rendre applicable les normes établies par l'ISO et le CEN.
Elle existe depuis plus de 80 ans, on la connait notamment pour les normes NF et AFAQ.

fig 21 : Les logos de l'AFNOR

D'autres associations oeuvrent pour la normalisation des transports, en voici une liste non-exhaustive.
- International Air Transport Association (IATA)
- International Union of Railways (UIC)
- OpenTravel Alliance (OTA)
- Organization for the Advancement of Structured Information Standards (OASIS)

II.2 : Les modèles existants.

Il existe de nombreux modèles pour les systèmes de transport, que ce soit des modèles scientifiques, à but commercial ou même pour des jeux et jouets (modélisme, simulateurs PC). Il est néanmoins très difficile d'obtenir ces modèles car ils sont souvent protégés par des licences commerciales ou sont confidentielles pour certaines applications. Les modèles retenus pour ces études sont tirés du site http://www.transportmodeller.com/. Le but de ce site est de montrer une approche du problème de modélisation dans le domaine du transport. Dans cette étude, nous allons approcher le modèle selon 3 points de vue:

- L'infrastructure routière et/ou de transport public. (le réseau en lui même).
- Simuler le choix d'un utilisateur entre les différents services de transport.
- L'évolution du trafic au fil des ans.

1-Modèle de l'infrastructure (assignement model):

On peut spécialiser ce modèle en 2 catégories : le transport privé et le transport public.

1.1-Transport privé (voitures):

On utilise un graphe pour la représentation des routes (arcs), elles sont divisées par zones géographiques (noeuds). Chaque zone géographique contient un point central (centre de gravité) qui sert pour le calcul de la densité de trafic.

On associe à chaque tronçon (arc) certaines informations comme:
- la longueur
- La vitesse maximale
- La capacité
- Le trafic moyen
- La vitesse moyenne (dépend du trafic)

On peut se servir de ce modèle indépendamment pour visualiser et résoudre certains problèmes de congestion, pour le choix de rénovation, création ou fermeture des routes. L'organisme de gestion doit effectuer des enquêtes pour définir les valeurs ci-dessus:

1.2-transport public (métro, tramway, train, etc...):

On peut utiliser le même type de modèle dans le cas des transports publics, en utilisant un système de graphe auquel on modifierait certaines informations du tronçon, notamment en y insérant des données sur la tarification pratiquée (le modèle proposé, indique une tarification en fonction de la distance parcourue, ce qui est très peu pratiqué en France).
Les faiblesses de ce modèle sont analogues aux faiblesses du modèle routier car on ne prend pas en compte le choix de l'utilisateur qui pourrait choisir un transport privé si les inconvénients du transport public augmentaient.

1.3-Forces & faiblesses du modèle:

Ce modèle ne peut être utilisé que pour des villes de tailles réduites car l'influence des autres moyens de locomotion qui peuvent être mis en place en parallèle, ne sont pas pris en compte dans ce modèle.

Exemple : *fermeture d'une route pour travaux, est-on sur que la quasi-totalité des usagers habituels de ce tronçon sera divisée sur d'autres parcours? Si des usagers ne préféreraient pas choisir le tramway ou le métro comme moyen de locomotion?*

1.4-Conclusions sur ce modèle:

Bien qu'il puisse fonctionner de manière indépendante, ce modèle n'est pas assez complet pour effectuer un générateur fiable car il ne prend pas en compte les paramètres humains, ainsi que d'éventuelles évolutions du contexte démographique. Cependant, il est très efficace comme modèle d'infrastructure en l'associant avec d'autres modèles.

2-Modèle du choix (choice model):

1.1-Le modèle arithmétique:

Ce modèle comble une des faiblesses du modèle précèdent car il prend en compte le choix de l'utilisateur entre le transport privé et le transport public.
Ce choix se fait avec l'aide des informations suivantes:

- les temps d'attente aux stations.
- les temps de déplacement entre le point de départ et la première station.
- les temps de déplacement entre le point d'arrivée et la dernière station.
- les durées de trajet.
- Le prix du voyage.

Le concept d'utilité a été inventé par des économistes pour modéliser l'utilité ou le bénéfice que pourrait avoir un utilisateur selon un choix donné.
Dans le contexte de ce modèle on obtient une fonction suivante:
U(m) = b(m) + b1*IVT(m) + b2*PRI(m) + b3*WALK(m) + b4*WAIT(m) + b5*IC(m)

Où:
m : moyen de transport choisi pour un trajet donné.
U(m) : l'utilité de voyager par ce moyen de transport.
b(m): paramètre constant du moyen de transport (cote auprès de l'utilisateur).
b1,b2,b3,b4,b5 : poids de chaque paramètre (peut être négatif).
IVT(m) = le temps accumulé passé dans le moyen de locomotion.
PRI(m) = le prix du moyen de locomotion (prix du billet, essence, péage, parking).
WALK(m) = le temps de marche entre les différentes stations de bus ou de train pour les transports publics, ou le temps de marche jusqu'au lieu de parking du véhicule personnel.
WAIT(m) = temps passé à attendre lors du voyage.
IC(m) = nombre de transferts entre les différents moyens de locomotion.
Une grande valeur de U (m) indique une « utilité » du moyen de transport importante.

Calcul de la probabilité du choix de chaque moyen de transport pour un trajet donné:
Pour calculer ces probabilités, il faut tout d'abord effectuer une étude préliminaire pour fixer les poids de la fonction d'utilité présente ci-dessus. Ensuite on peut calculer les probabilités de choix avec la relation suivante.

$$P(m)=\frac{e^{U(m)}}{\sum e^{U(m)}}$$

1.2-Conclusions sur ce modèle:

Cette modélisation du choix d'un utilisateur est intéressante à utiliser en complément du modèle étudié précédemment, car elle permet de simuler le trafic en fonction du nombre de trajets demandé et surtout d'établir les proportions de voyageurs sur les différents modes de transports. On peut l'utiliser pour la prospection et la création de nouvelles lignes pour les compagnies de transports en commun. Mais il nous manque toujours l'évolution du trafic au cours du temps.

3-Modèle de gravitation (gravitation model):

Selon la loi de la gravitation universelle, la norme de la force d'attraction est dépendante de la distance et de la masse des deux corps.
Le modèle de gravitation reprend ce principe, en quantifiant le trafic entre deux zones grâce à la distance les séparant ainsi qu'à leur poids (population, commerces, centres d'intérêts divers).
Car deux zones importantes, au niveau démographique et à proximité, ont une tendance à produire du flux de personnes entre ces deux zones.
Ce modèle permet d'ajuster le trafic pour le système global d'année en année en faisant varier les attributs des zones géographiques.

La qualité de ce modèle, est directement liée à la qualité de son calibrage. Il doit être constant sur toutes les zones étudiées. On obtient donc en combinant ces 3 modèles un modèle assez puissant pouvant anticiper des problèmes de transport et ainsi les réguler au mieux.

II.3-Les études existantes:

1-Les simulateurs grand public :

Le but de cette partie est d'énumérer quelques simulateurs existants, dans le domaine du chemin de fer.

-Microsoft Train Simulator http://www.microsoft.com/games/pc/trainsim.aspx

Jeu de gestion et de conduite de train, édité par Microsoft, il est actuellement le simulateur grand public le plus connu au monde.

-Trainz Simulator http://www.ts2009.com/

Logiciel de construction et de gestion ferroviaire, développé par Auran. Il est très complet et bénéficie d'une communauté importante ce qui permet d'avoir beaucoup d'extensions disponibles.

-Virtual Railroad 3 http://www.simplytrains.com/pages/vr3/vr3.htm

Logiciel de construction ferroviaire, développé par le studio allemand Trend c'est l'un des simulateurs le plus complet. Néanmoins il est aussi l'un des plus difficile à utiliser.

-Visual-tm http://www.visual-transport-modeller.com

Simulateur reprenant le modèle étudié ci-dessus, on peut l'utiliser dans la gestion d'un réseau de transport public dans le domaine de l'aide à la décision.

2-Les publications :

De nombreuses publications scientifiques sur le domaine du transport ont été publiées. Nous avons recherché ces différentes publications pour les référencer et nous permettre de voir ce qui a déjà été fait dans ce domaine. Celles-ci sont représentatives des différentes études trouvées:

- Uniformisation des Méta-Modèles nationaux :

Cette étude **[DE JONG et coll.]** pose les bases pour l'uniformisation des Méta Modèles. En effet, les modèles de transport de passagers et de marchandises sont différents dans chaque pays de l'Union Européenne. Cette publication est donc basée sur l'étude de cinq modèles nationaux sur le transport de personne, afin d'établir un méta-modèle de transport unique européen.

- Identifications des problèmes de logistique :

L'objectif de cette publication **[LAPORTE]** est d'identifier les différents problèmes de logistiques pour l'établissement d'un planning pour le transport de marchandises, afin d'obtenir une optimisation de l'utilisation des réseaux. En effet, l'objectif est de résoudre ces problèmes afin de pouvoir utiliser le plus de train possible sur le même réseau.

- Modèle du Shinkansen train :

Cette publication **[TANABE et coll.]** propose un module numérique pour l'analyse des interactions dynamiques entre le «Shinkansen train» (train japonais pour le transport de marchandises) et les structures disponibles. Il permet ainsi de déterminer des contraintes pour ce train lors du passage sur certaines infrastructures comme les ponts ou les tunnels.

- Optimisation du planning :

Cette étude **[ZWANEVELD et coll.]** décrit les problèmes de routage des trains à travers les stations de chemin de fer. Il propose des algorithmes pour optimiser les passages des trains dans le réseau. Elle décrit également comment gérer les problèmes qui peuvent subvenir pendant un trajet, comme par exemple une panne.

- Elasticité de la demande :

Cette étude **[*KUO* et coll.]** décrit les planifications des départs de train selon l'élasticité de la demande afin d'optimiser le nombre de train à utiliser. Elle propose des études statistiques pour déterminer combien de personnes peuvent prendre le train dans une ville selon ses caractéristiques (nombres d'habitants, autres moyens de transport).

- Interactions entre trains et ponts :

Cette publication **[HE XIA et coll.]** décrit les interactions entre les trains et les ponts. Elle parle spécifiquement d'un type d'infrastructure, et peut permettre de définir les paramètres pour le passage du train. Cette étude s'appuie sur une étude pratique et théorique, et peut permettre d'optimiser les contraintes de passage, tout en conservant la sécurité.

- Modélisation de la circulation :

Cette étude **[XIE et coll.]** est celle qui se rapproche le plus de notre objectif. Elle explique comment faire circuler un train sur un sillon, selon la configuration du réseau et les infrastructures disponibles, et suivre ainsi un planning en l'optimisant.

Pour conclure, on voit ici que la question de la modélisation du transport se pose dans de nombreux pays pour permettre d'optimiser la circulation sur les réseaux ferrés, éviter de développer les réseaux déjà existants et ainsi optimiser les coûts. Egalement, de nombreux domaines sont concernés.

III Etude du modèle de sillon de RFF

III.1 Présentation de RFF.

RFF (**Réseau Ferré Français**) est depuis 1997 propriétaire et gestionnaire des infrastructures ferroviaires, il hérite en contrepartie de la dette de la SNCF et signe avec l'opérateur une convention pour la gestion de trafic et l'entretien du réseau.

fig 22: Logo RFF

Voici quelques dates importantes dans l'histoire de RFF:
En 2000, le RFF mène son premier projet de la LGV (Ligne Grande Vitesse) Est Européenne.
En 2001, le RFF obtient le feu vert pour la ligne ferroviaire Lyon-Turin.
En 2003, il est chargé d'un autre mission: l'attribution des capacités allouées par le RFF.
En 2007, inauguration de la LGV Est européenne, réalisation d'un nouveau record de vitesse à 574,8 km/h

Ses objectifs sont de :
- Organiser la circulation des trains (15 000 par jour)
- Augmenter le potentiel d'attractivité du réseau
- Développer de nouvelles lignes ferroviaires
- Optimiser et valoriser le réseau actuel

III.2 Etude du modèle donné:

Pour une explication plus simple et une meilleure lisibilité, le modèle sera séparé en 3 parties. Le modèle complet se trouve en annexe 1.

1-Modélisation des sillons et des missions:

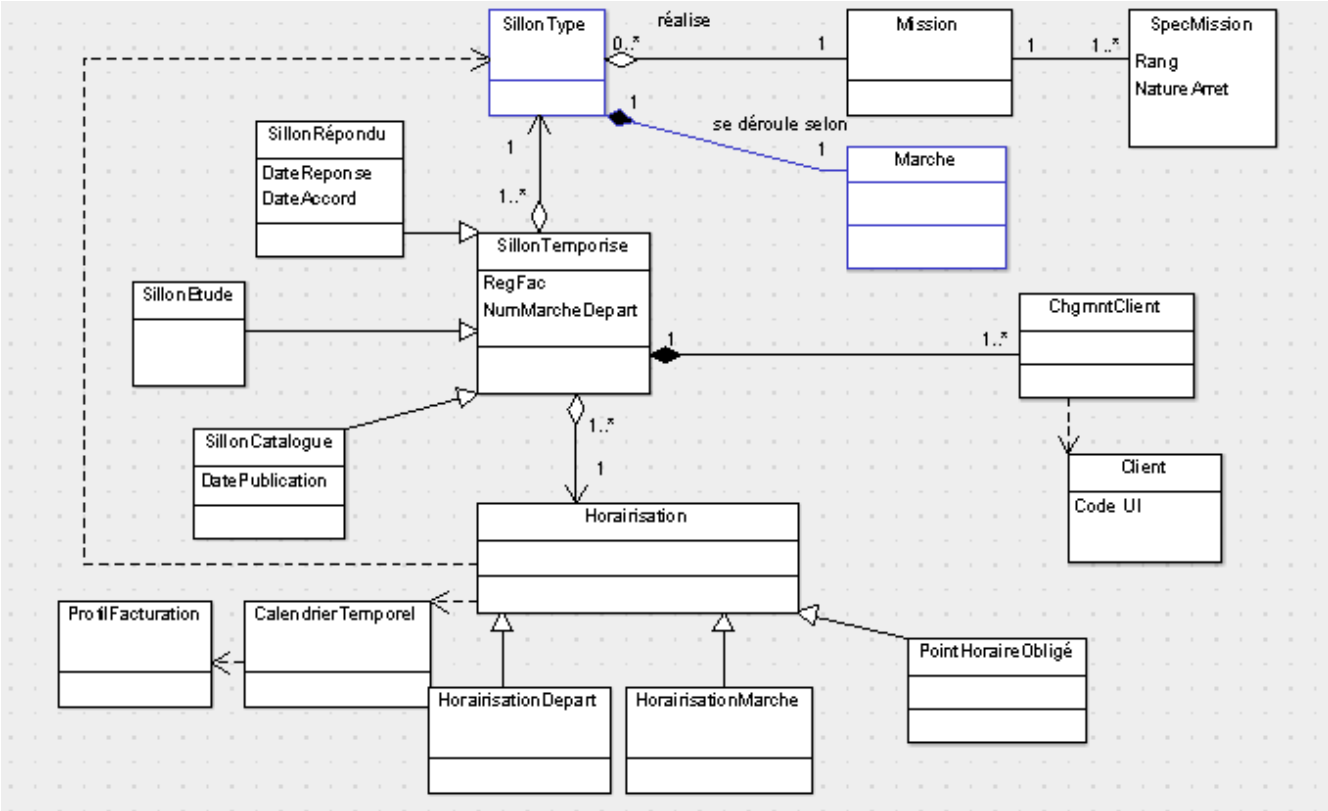


fig 23 : Modélisation des sillons

Dans le domaine du transport ferroviaire, un sillon représente la capacité d'infrastructure requise pour faire circuler un train donné d'un point à un autre à un moment donné. Un sillon est modélisé par la classe SillonType. Elle permet de factoriser des sillons sur la base d'une horairisation pour représenter des sillons cadencés (qui se reproduisent plusieurs fois dans la journée avec une translation de temps). Le sillon possède une mission, un sousGrapheParcours (le graphe de parcours du train sur le réseau) et une marche.

Premièrement la mission décrit le politique de desserte avec les différentes informations comme les arrêts et la destination. Elle possède plusieurs SpecMission. Ces specMission représentent les arrêts de la mission. Une specMission possède un rang pour retrouver l'ordre des passages et une nature de l'arrêt. Une specMission fait référence à un PointRemarquable du réseau. Cette partie sera expliquée en détails dans l'infrastructure.

Ensuite, sousGrapheParcours représente le sous graphe du réseau d'infrastructure dans lequel le train circule. Les graphes seront plus détaillés dans la partie Graphe infrastructure.

On modélise un sillon correspondant à un sillon type à un horaire précis grâce à SillonTemporise. Il correspond donc à un sillonType et possède une horairisation. L'horairisation dépend du calendrierTemporel. Cette classe peut être spécialisée pour correspondre à un départ, une marche ou des horaires obligés, c'est à dire combien de retard peut avoir un train sans retarder les autres programmés.
Les horairisations correspondent à un ProfilFacturation, c'est à dire qu'à chaque horaire correspond un prix qui sera facturé aux clients qui réservent le sillon pour le train.

Enfin, la classe SillonTemporise peut être spécialisée pour représenter l'état d'avancement d'une réservation de sillon. En effet, lorsqu'un client de RFF veut effectuer une demande, il va choisir dans les sillonscatalogues: ce sont les sillons qu'un horairiste (personne qui enregistre les sillons et les réparti entre les clients) a enregistré dans un catalogue et qui correspondent aux sillons libres. Si cela ne lui convient pas, il peut demander un sillon particulier non présent dans le catalogue. Il y a donc une création d'un sillonEtude. Ensuite, lorsque la demande de sillon devient valide, on a donc un SillonRepondu qui possède une date de réponse et une date d'accord.

Les sillons possèdent également des marches.

2-Modélisation des *marches*:

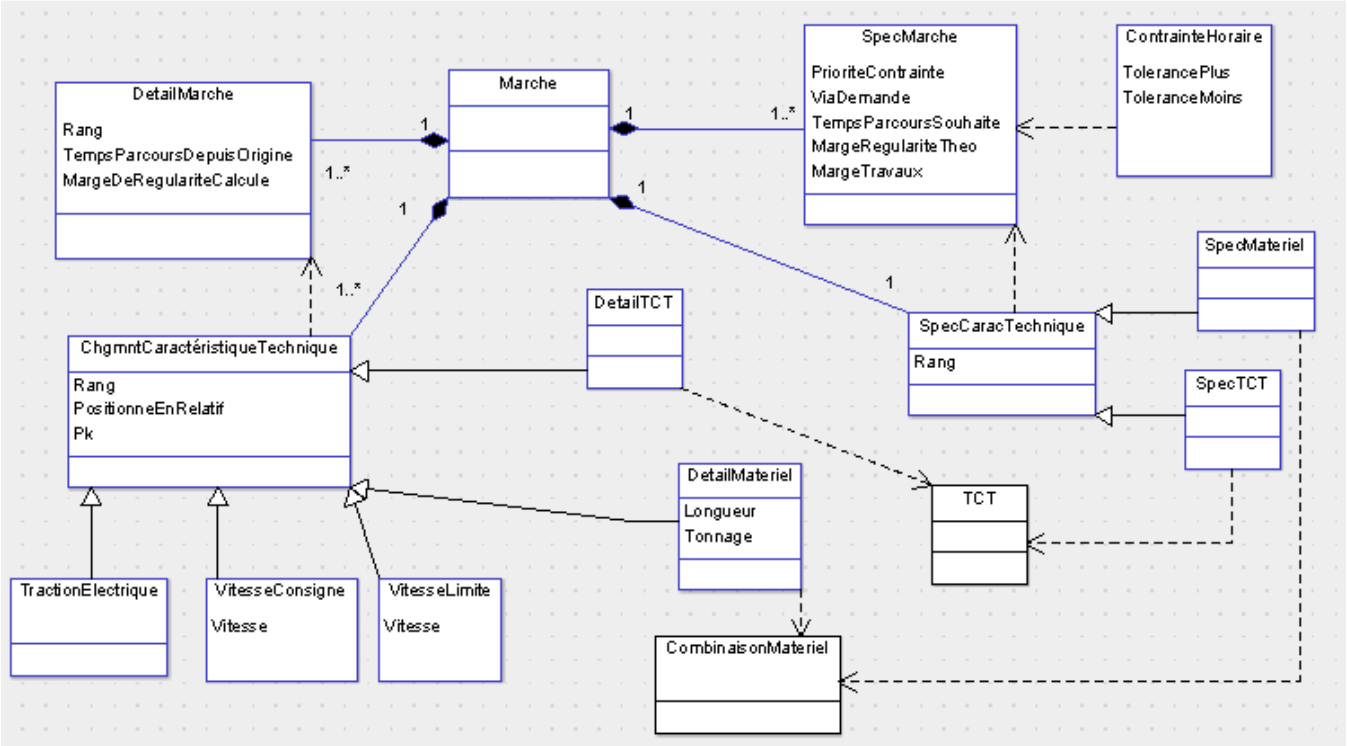


fig 24: Partie du diagramme UML concernant les Marche

Une marche permet de calculer les accélérations et freinages, et donc les vitesses d'un train sur un parcours. La marche possède des detailsMarche (détails de la marche), des SpecCaracTechnique (caractéristiques techniques à respecter), des chgmntCaracteristiqueTechnique (changements pendant le parcours du train), des spécificités de techniques et des specMarches.

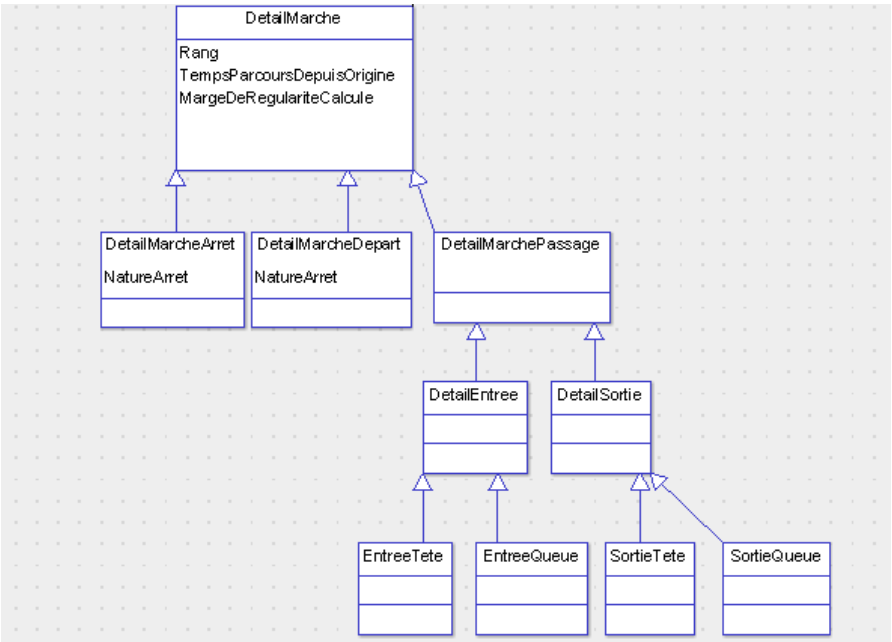


Fig 25: Partie du diagramme UML sur les DétailsMarche

Premièrement, les DetailsMarche décrivent les différentes étapes de marche du train. Cela peut être un arrêt (DetailMarcheArret), un départ (DetailMarcheDepart), ou un passage (DetailPassage) comme par exemple, le passage d'un tunnel ou d'un pont. On a une spécialisation pour chaque type de detailMarche. De même, on peut spécialiser l'entrée et la sortie du passage (DetailEntree et DetailSortie). Ils peuvent également être spécialisées pour représenter l'entrée de la tête du train et l'entrée de la queue du train (il en est de même pour la sortie).
Un détail fait référence à une ressourceInfra qui représente l'infrastructure correspondante. Par exemple, un arrêt peut se faire dans une gare et un passage dans un tunnel, un pont ou une courbure.
Il fait également référence à un horaire (horairisation).
La classe possède un rang pour connaitre l'ordre de passage pour les détails, un temps de parcours depuis l'origine et une marge de régularité calculée (c'est la marge qui permet de faire face aux aléas qui peuvent avoir lieu et ainsi éviter une trop grande perturbation du planning initial).

Ensuite, le *ChgmntCaracteristiqueTechnique* représente un changement des caractéristiques techniques du train lors d'un passage sur un *détailMarche*. Cela peut être un moyen de traction particulier (*tractionElectrique*), une vitesse limitée (*vitesseLimite*) ou obligatoire(*vitesseConsigne*), un détail sur le matériel(*DetailMateriel*) comme la longueur ou le tonnage requis à un endroit du réseau ou un type de Convoi-Traffic(DetailTCT). En effet, une ligne peut être réservée à un type de transport (transport de marchandises ou de voyageurs).

Le détailMatériel quand à lui, possède une longueur et un tonnage. Cette classe fait référence à une combinaison de matériel (*CombinaisonMateriel*). Cette classe représente les différentes combinaisons matérielles à utiliser pour un train. Elle fait référence à un moteur (*EnginMoteur*), et un engin de calcul (*EnginDeCalcul*) avec un indice de composition. Nous n'avons toutefois que peu d'informations sur la correspondance exacte de ces classes.

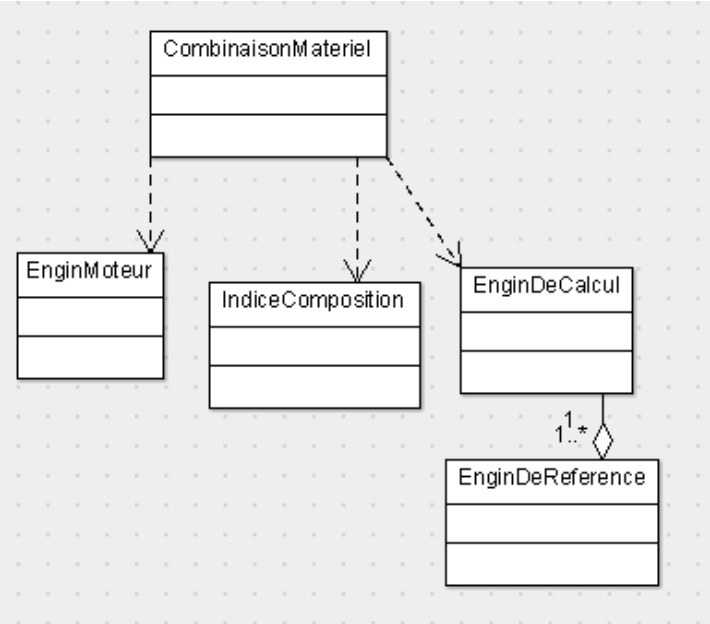


fig 26: Partie du diagramme UML sur la CombinaisonMateriel

La SpecMarche représente les spécificités de la marche. Elle possède une priorité de contrainte (pour savoir quel est la plus importante), un temps de parcours souhaité, une marge de temps de régularité et une marge de temps lors des travaux. On peut donc trouver les retards minimums possibles pour ne pas perturber l'ensemble des marches.

Finalement, les *SpecCaracTechniques* correspondent aux caractéristiques globales du train. Une marche n'en possède qu'une seule. Cela peut être une spécificité TCT (*SpecTCT*), ou une de matériel (*SpecMateriel*).

3-Modélisation du graphe d'infrastructure:

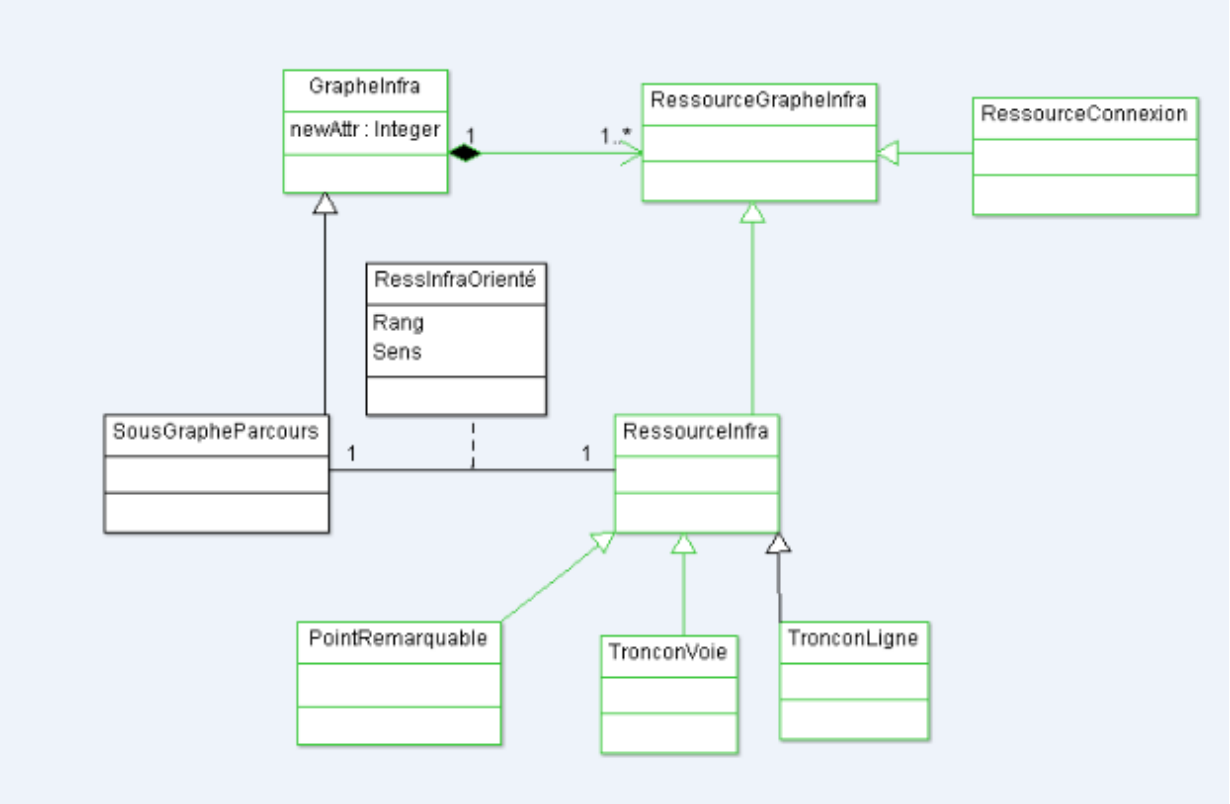


fig 27: Partie du diagramme UML sur les infrastructures

Le *GrapheInfra* correspond au réseau des infrastructures de RFF. Il possède des graphes de ressources (*RessourceGrapheInfra*). Ces ressources peuvent être des infrastructures ou des connexions de ressources. Les ressources (*RessourceInfra*) sont les infrastructures du réseau. Cela peut être un *PointRemarquable*, comme une gare, un tunnel, ou un pont. Cela peut être également un tronçon de voie ou de ligne (*TronConVoie* et *TronconLigne*). Elle est reliée à un sous-graphe de parcours (*SousGrapheParcours*) avec une classe d'association *RessinfraOriente* qui représente une orientation pour les ressources (avec un rang et un sens).

SousGrapheParcours représente un sous graphe de parcours, c'est à dire un graphe correspondant à un sillon.

III.3 Eléments manquants dans le modèle et critique de l'ensemble:

Après étude du modèle et de la documentation fournie par RFF expliquant le fonctionnement du réseau ferré, nous avons trouvé certains défauts à ce modèle.

Premièrement, il n'y a aucun attribut dans les classes. Il est donc difficile de comprendre exactement à quoi correspondent certaines classes comme par exemple, la classe indiceComposition ou quelles sont les caractéristiques des moteurs importantes dans la modélisation du réseau ferré.

Deuxièmement, il semble y avoir certaines erreurs comme la relation entre RessourceInfra et SousGrapheParcours, qui est redondante. Une autre modélisation permettrait de réduire le modèle.

Certaines notions sont également manquantes par rapport à la documentation qui nous a été fournie:

Dans les *ChgmtCaracteristiquesTechnique*, une spécialisation permet de décrire les informations sur la TractionElectrique, mais il manque la traction diesel.

La notion de régime d'un sillon (jours de circulations autorisés) n'apparait pas.

La notion de marche économique (marche pour réduire la consommation, par exemple une vitesse conseillée) n'est pas représentée.

Les règles de sécurité arbitrant les conflits ne sont pas explicitées.

Les sillons de dernières minutes n'existent pas dans le modèle actuel.

De plus, certaines spécifications de classe semblent abusives. Par exemple, la spécialisation des horaires, ou les détails de marche.

Ce modèle est toutefois très modulable. Ainsi, différentes caractéristiques de marche peuvent être ajoutées très facilement. Ce modèle prévoit donc la possibilité pour celui-ci d'évoluer.

IV Réalisation d'une simulation d'un réseau ferré

IV.1 Cahier des charges :

Une fois l'étude du modèle réalisée, il nous a été demandé de réaliser une simulation utilisant une version simplifiée de celui-ci, dans le but d'appliquer nos connaissances, et de valider nos conclusions. N'ayant pas d'instruction particulière, nous avons donc décidé de fixer le cahier des charges suivant :

-La simulation sera représentée en graphe macroscopique (description agrégée du réseau ferré). Les gares représenteront des nœuds, et les voies des arrêtes.

-L'application sera modulable, c'est à dire que l'on pourra ajouter d'autres éléments facilement à notre modèle.

-Une gestion des collisions sera réalisée pour éviter que les trains se percutent.

-Une gestion des pannes. Un train pourra tomber en panne, puis être redémarrer. Les autres trains devront réagir en conséquence pour qu'il n'y ait pas d'accident.

-Pour les changements de caractéristiques, la vitesse des trains pourra être modifiée sur un point particulier du réseau.

-Il n'y aura pas d'optimisation de planning de trains. Il sera choisi aléatoirement, et de façon à générer des conflits entre trains.

-L'implémentation sera faite en C# avec Visual Studio 2010 avec une interface graphique en GDI+. Ce langage a été utilisé pour sa facilité de création d'interface graphique simple.

-Le modèle sera simplifié. Beaucoup de caractéristiques sur le matériel de train nous étant inconnues, il était inutile d'implémenter ses classes.

IV.2 Modèle retenu pour notre simulation :

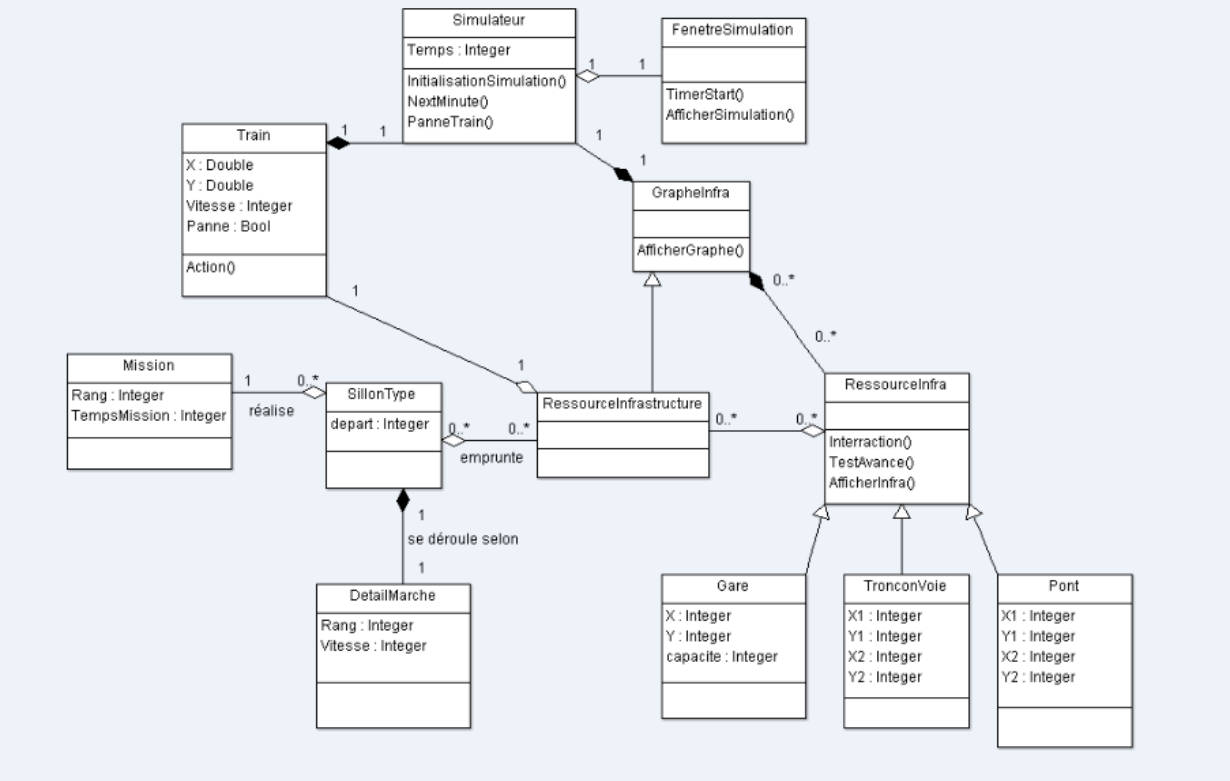


fig 28: Modèle de la simulation

Ce modèle est celui que nous avons utilisé. Il s'inspire du modèle existant de RFF.

La partie graphe infrastructure a été simplifiée, puisque la classe RessourceConnexion a été supprimée. Les seuls éléments d'infrastructure utilisés seront les gares, les tronçons de voies, des ponts et des tronçons de voies. La gare possède une capacité, c'est-à-dire un nombre maximum de trains que peut accueillir celle-ci en même temps. Le pont est un élément ne possédant qu'une seule voie : deux trains ne peuvent donc pas se croiser et un aiguillage permet de changer le train qui doit passer. Le TronconVoie quand à lui correspond à deux voies côte à côte pour permettre de faire passer deux trains de sens opposé en même temps. Ces classes dérivent d'une classe abstraite, RessourceInfra. Il peut donc être rajouter pour une simulation plus complexe des classes dérivées supplémentaires correspondant à une autre ressource d'infrastructure. La généricité permet donc d'agir de la même façon sur les différents types d'infrastructures. Il y a donc une grande modularité.

Les classes SillonType et SillonTemporise ont été fusionnées. En effet, nous n'en avions pas besoin pour remplir notre cahier des charges, puisqu'il n'y a pas de persistance des données. Le départ permet de savoir à quel moment doit partir le train.

Les missions ont également été simplifiées. Cette classe possède un rang, pour savoir à quel moment aura lieu la mission. Elle correspond pour notre simulation à des arrêts dans une gare.

Pour la marche, et les détails de marche, la seule caractéristique utilisée sera la vitesse qui pourra évoluer selon la ressource traversée, les autres caractéristiques étant trop complexes et nous étant inconnues. Le rang permet de savoir, comme pour les missions, à quel endroit à lieu le changement de vitesse.

La classe principale sera le simulateur. Ce simulateur permet d'initialiser la simulation, et de lancer chaque itération sur les trains. La gestion des trains se fera donc grâce à cette classe. Elle déplacera les trains chacun leur tour sur le réseau, et pourra mettre en panne ou réparer un train.

La classe FenetreSimulation représente la partie graphique du simulateur. Cette partie est indépendante du reste. La modification de la vue se fait donc seulement par cette classe, pour respecter le principe du MVC (Modèle Vue Contrôleur). Cette classe utilise la bibliothèque GDI+.

IV.3 Présentation de la simulation :

1-Gestion de l'affichage:

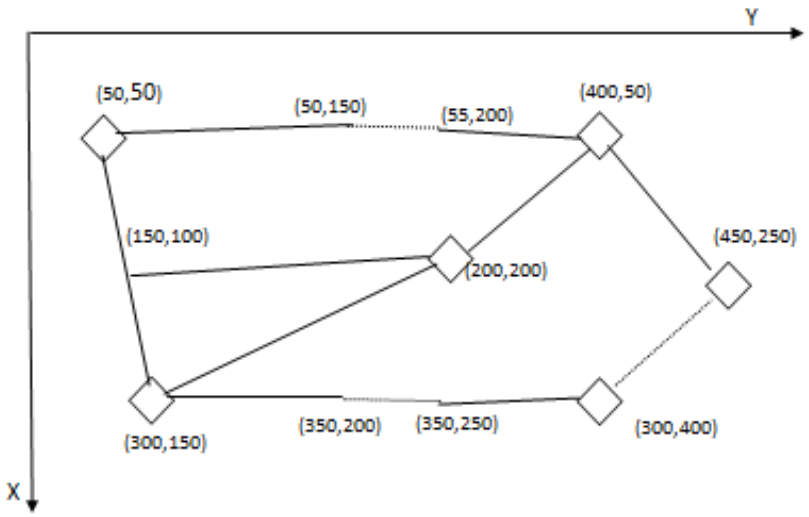


fig 29: Plan de la simulation

Les infrastructures et les trains sont représentés sur un plan à deux dimensions. Nous avons choisi arbitrairement un modèle de voies ferrées (représenté ci-dessus). Il possède tous les types d'infrastructures définies.

Pour réaliser une simulation avec un autre modèle, il suffit de dériver notre classe simulateur et de ré-implémenter la méthode « InitialisationSimulation ». Un buffer d'image a été utilisé pour éviter un scintillement de l'image lorsque l'affichage se met à jour.

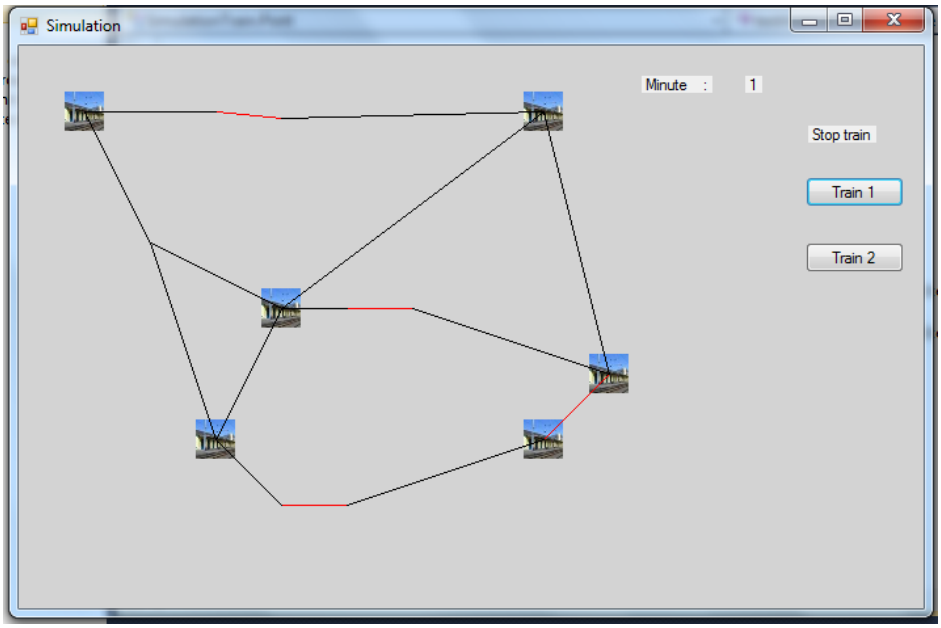


fig 30 : Affichage du réseau

Une fois le simulateur lancé, le graphe s'affiche. Les gares sont représentées par une icône, les voies par des traits noirs, et les ponts en rouge. Un label permet d'afficher le nombre de minutes écoulées depuis le début de la programmation, pour contrôler si le planning du train est respecté. A droite, des boutons s'affichent avec un numéro de train. Ces boutons permettent de faire tomber un train en panne. Le nombre de boutons se fait automatiquement avec le nombre de train choisit pour la simulation.

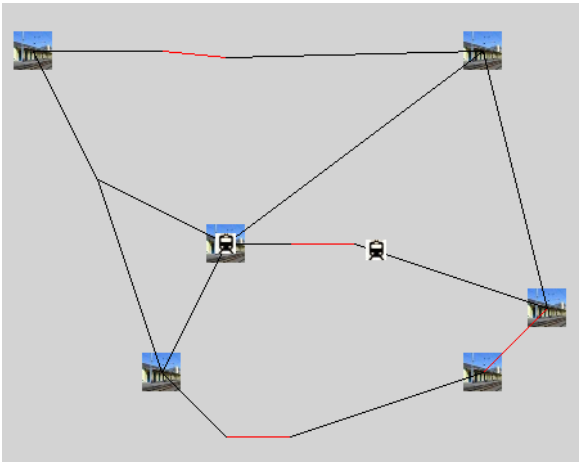


fig 31: Affichage du déplacement des trains

Un timer a été initialisé pour être déclenché toutes les demi-secondes. Celui- ci va permettre de déplacer le train sur le plan selon le planning. Les déplacements se font grâce à des vecteurs directeurs. Le train va donc se déplacer vers l'infrastructure suivante. Les trains s'affichent par une icône. Celle-ci change de couleur quand il est en panne.

2-Gestion des collisions

Maintenant que les trains se déplacent correctement, il faut donc gérer les collisions. En effet, si un train tombe en panne ou passe sur un pont, il faut éviter que les deux trains se percutent.

Ils existent plusieurs cas de collisions possibles :

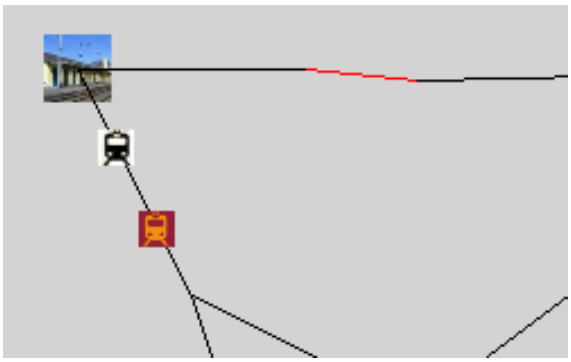


fig 32: Gestion d'une collision sur une même voie

Premièrement, si un train tombe en panne sur une voie, ou un pont, il faut que les trains le suivant s'arrêtent. Nous avons défini une marge de sécurité correspondant à la distance parcourue à 4 kilomètres avec la vitesse maximale du train. Le train suivant va redémarrer quand le train en panne sera réparé.

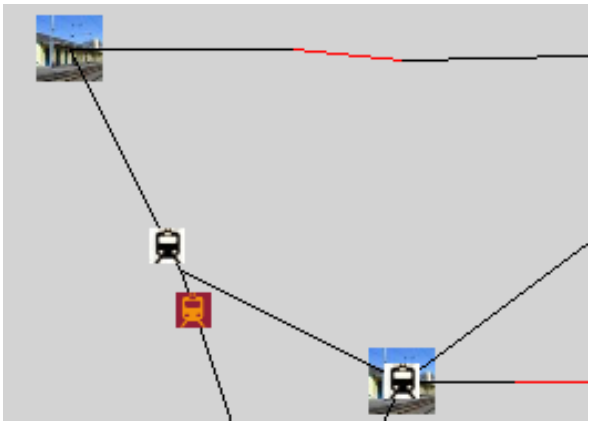


fig 33: Gestion de la collision sur des voies différentes

Un autre cas possible est qu'un train s'arrête au début d'une voie. Les trains le suivant et se trouvant sur la voie précédente détecte donc cet arrêt, et ne continue pas sur l'infrastructure suivante.

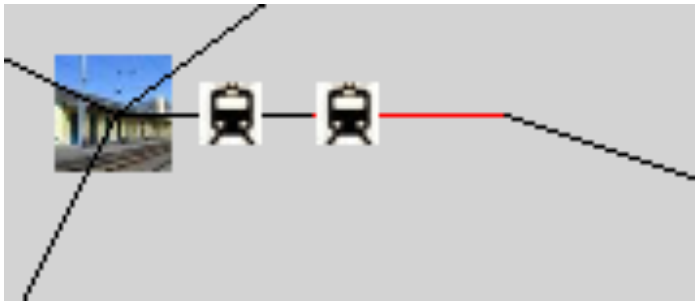


fig 34: Gestion d'une collision sur un pont

De manière similaire, la détection doit être faite avant le passage sur un pont d'un train. Il doit donc vérifier qu'aucun train ne se trouve dans l'autre sens. Il peut toutefois passer sur le pont si un train circule sur le pont dans le même sens.

Un autre cas est la panne sur un pont. Il est dangereux pour un train et ses passagers de rester bloqué sur un pont. Pour éviter certains accidents, un train ne traversera pas un pont si un train est bloqué dessus, même si celui-ci circule dans le même sens.

Pour conclure, on peut dire que la réalisation de notre simulation nous permet de vérifier les remarques que nous avions faites sur le modèle initial. Ainsi, celui-ci est très modulable, et l'ajout de nouvelle infrastructure se fait de manière simple. Bien que nous ayons seulement utilisé la vitesse comme caractéristique de marche, il est aisé d'en rajouter pour ainsi complexifier le modèle.

Cela confirme aussi que les règles de sécurité ne sont pas présentes. La marge de sécurité a été écrit en dur, et ne peut pas changer selon le type d'infrastructure parcouru.

Conclusion

Pour faciliter la compréhension de l'outil de modélisation utilisé par RFF, nous avons réalisé un tutorial, résumant les méthodes pour réaliser des diagrammes UML. Il pourra être utilisé pour un apprentissage rapide de ce logiciel. En reprenant l'analyse des points positifs et négatifs effectuée précédemment, nous pouvons donc répondre par l'affirmative à la question posée dans l'introduction qui remettait en cause l'utilisation de ce logiciel pour ce type de projet.

L'analyse du modèle proposée par RFF, a été faite en détaillant chaque partie. Elle nous a permit de relever l'existence de redondances dans les données utilisées ainsi que certaines erreurs dans les relations entre certaines classes. Mais notre remarque la plus importante porte sur le manque de certaines informations qui peuvent paraître primordiale dans le contexte actuel, surtout dans le domaine de la sécurité et de l'économie des ressources.

Notre plus grande difficulté est survenue sur la partie de recherche de normes et de modèles existants. En effet cette méthodologie étant nouvelle pour nous, les premiers résultats ont été infructueux. Après avoir acquis certaines méthodes de recherche, nous avons pu réaliser une étude plus précise, et ainsi remarquer que de nombreuses recherches sur la modélisation ont déjà été effectuées quelque soit la région du globe, dans de nombreux domaines.

Pour valider nos résultats, nous avons réalisé une simulation fonctionnelle, remplissant entièrement le cahier des charges que nous avons réalisé, et qui pourrait être complété pour complexifier celui-ci.

L'ensemble de notre documentation produite peut donc servir pour la recherche et une proposition d'un nouveau modèle plus performant, permettant de corriger les défauts que nous avons remontés.

Bibliographie :

p17 : [DE JONG et coll.] : Gérard de Jong, Hugh Gunn, Moshe Ben-Akiva, A meta-model for passenger and freight transport in Europe Original Research Article, *Transport Policy Volume 11 Issue 4*, *Pages 329-344* Octobre 2004.

p17 : [LAPORTE] : Teodor Gabriel Crainic, Gilbert Laporte, Planning models for freight transportation, *Journal of Operational Research Volume 97 Issue 3*, *Pages 409-438* 16 March 1997.

p17 : [TANABE et coll.] : M Tanabe, H Wakui, N Matsumoto, H Okuda, M Sogabe, S Komiya, Computational model of a Shinkansen train running on the railway structure and the industrial applications, *Pages 705-710*, 22 September 2003.

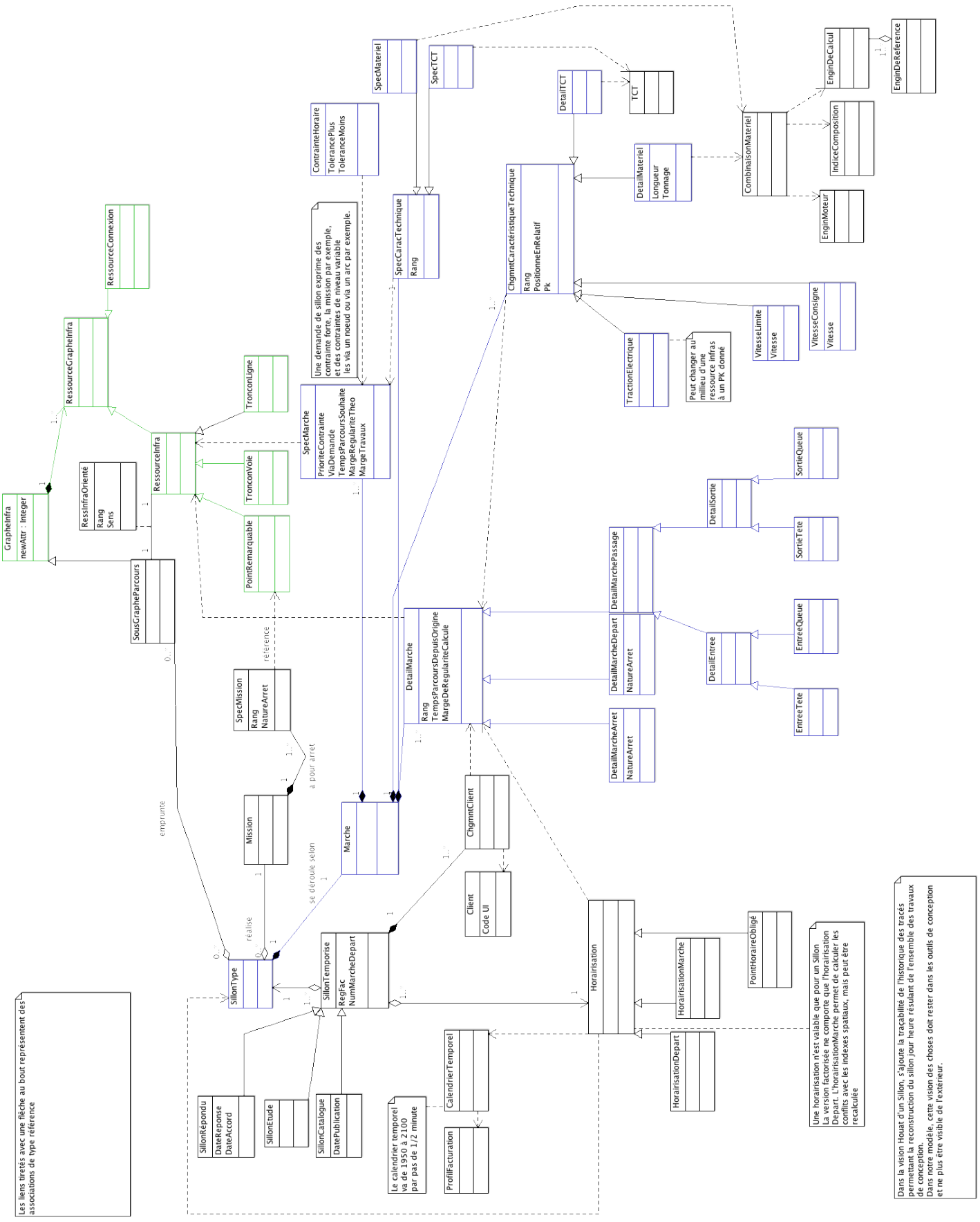
p17 : [ZWANEBEL et coll.] : Peter J. Zwaneveld, Leo G. Kroon, Stan P. M. van Hoesel, Routing trains through a railway station based on a node packing model, *European Journal of Operational* Research Volume 128 Issue 1, *Pages 14-33* January 2001.

p17 : [*KUO* et coll.] : April Kuo, Elise Miller-Hooks, Hani S. Mahmassani, Freight train scheduling with elastic demand, *Transportation Research Part E: Logistics and Transportation Review Volume 46 Issue 6*, November 2010.

p18 : [HE XIA et coll.] : He Xia, Nan Zhang, Dynamic analysis of railway bridge under high-speed trains, *Computers & Structures Volume 83 Issues 23-24, Pages 1891-1901,* September 2005.

p18 : [XIE et coll.] : Meiquan XIE, Baohua MAO, Tinkin HO, Zhenqi CHEN and Yun BAI*, Modeling Circulation of Trainnext term-Set with Multiple Routing*, 11 February 2010.

Annexes


SpecMateriel
SpecTCT
TCT
DetailTCT
ContrainteHoraire
TolerancePlus
ToleranceMoins
EnginDeCalcul
EnginDeReference
CombinaisonMateriel
IndiceComposition
EnginMoteur
DetailMateriel
Longueur
Tonnage
ChgmntCaractéristiqueTechnique
Rang
PositionneEnRelatif
Pk
SpecCaracTechnique
Rang
Une demande de sillon exprime des contrainte forte, la mission par exemple, et des contraintes de niveau variable les via un noeud ou via un arc par exemple.
RessourceConnexion
RessourceGrapheInfra
RessourceInfra
TronconLigne
TronconVoie
PointRemarquable
GrapheInfra
newAttr : Integer
RessInfraOrienté
Rang
Sens
SousGrapheParcours
SpecMarche
PrioriteContrainte
ValDemande
TempsParcoursSouhaite
MargeRegulariteTheo
MargeTravaux
TractionElectrique
Peut changer au milieu d'une ressource infras à un PK donné
VitesseLimite
Vitesse
VitesseConsigne
Vitesse
SortieQueue
SortieTete
DetailSortie
DetailMarchePassage
DetailMarcheDepart
NatureArret
DetailEntree
EntreeQueue
EntreeTete
DetailMarcheArret
NatureArret
DetailMarche
Rang
TempsParcoursDepuisOrigine
MargeDeRegulariteCalcule
SpecMission
Rang
NatureArret
référence
Mission
Marche
ChgmntClient
Client
Code UI
emprunte
réalise
se déroule selon
à pour arrêt
SillonType
SillonTemporise
RegFac
NumMarcheDepart
Horairisation
HorairisationMarche
HorairisationDepart
PointHoraireObligé
SillonRépondu
DateReponse
DateAccord
SillonEtude
SillonCatalogue
DatePublication
CalendrierTemporel
ProfilFacturation
Le calendrier temporel va de 1950 à 2100 par pas de 1/2 minute
Les liens tiretés avec une flèche au bout représentent des associations de type référence
Une horairisation n'est valable que pour un Sillon
La version factorisée ne comporte que l'horairisation Depart. L'HorairisationMarche permet de calculer les conflits avec les indexes spataux, mais peut être recalculée
Dans la vision Houat d'un Sillon, s'ajoute la traçabilité de l'historique des tracés permettant la reconstruction du sillon jour heure résultant de l'ensemble des travaux de conception.
Dans notre modèle, cette vision des choses doit rester dans les outils de conception et ne plus être visible de l'extérieur.