ENSICAEN
ÉCOLE NATIONALE SUPÉRIEURE D'INGÉNIEURS DE CAEN
& CENTRE DE RECHERCHE

Rapport Projet : ToolBarrierProject

MEUNIER Guillaume, Chef de projet
ADAM Rodolphe
BRUYERE Julien
GAZQUEZ Rémi
LOUIS Thomas
PIRES Alex
SLAVNIC NIKOLA

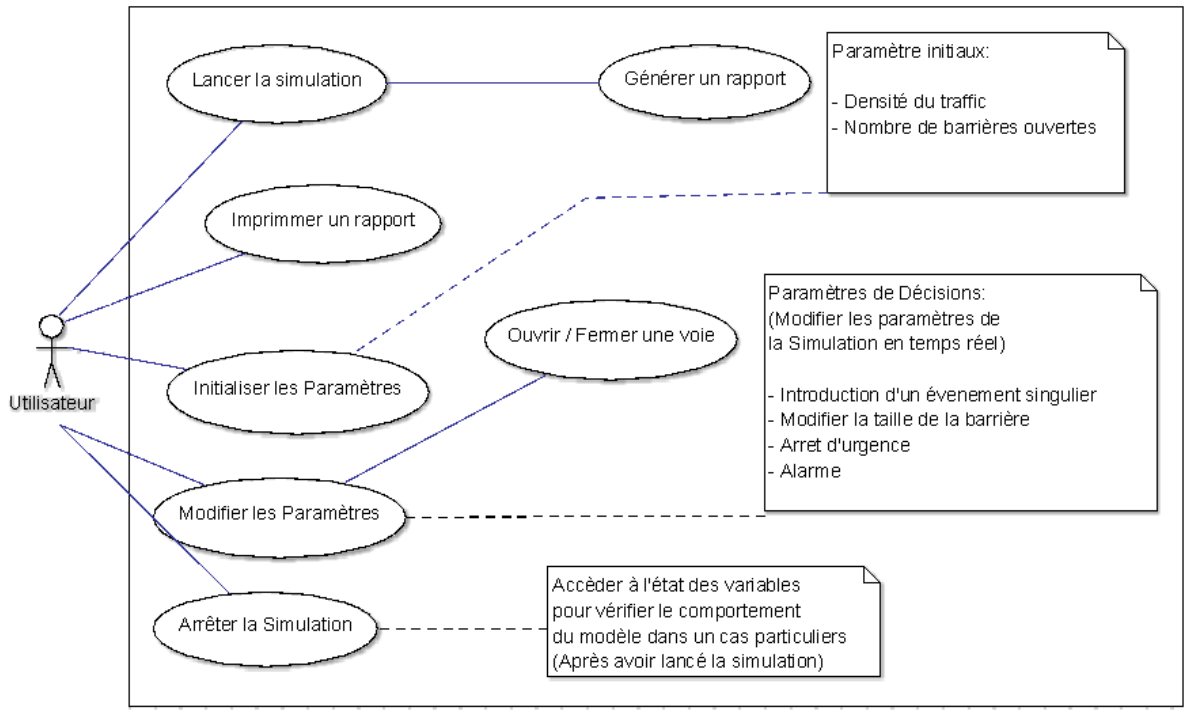
2ème année, Informatique
Année universitaire 2013/2014

ENSICAEN
6, boulevard Maréchal Juin – CS 45 053 – F- 14050 Caen Cedex 4
Tél. +33 (0)2 31 45 27 50
Fax +33 (0)2 31 45 27 60

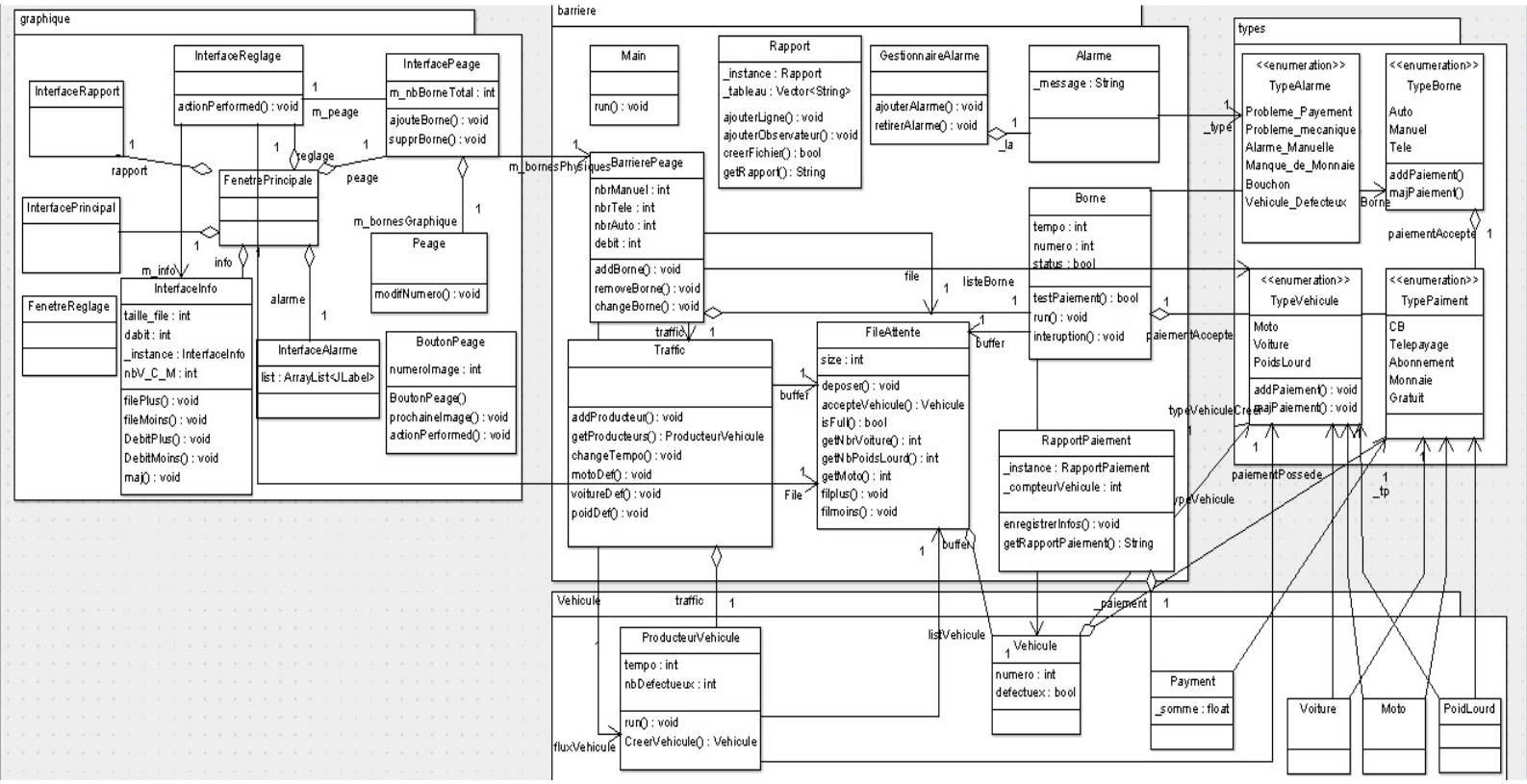
Table des matières

1 Modélisation de la solution

1.1 Diagramme des cas d'utilisation



1.2 Diagramme de classe



1.3 Explication du modèle

Dans un premier temps, nous initialisons une instance de la classe BarrierePeage qui connaît les différentes bornes, la file d'attente, et le trafic. BarrierePeage est appelée lors de la finalisation des premiers réglages par Fenetre_Reglages, et est réutilisé dans la Fenetre_Principale.

Chaque borne est un thread et accepte un véhicule de la file d'attente lorsque la borne est ouverte. Les bornes sont initialisées selon leur type, qui correspond à un temps moyen de passage. Dans l'interface graphique nous pouvons facilement ouvrir/fermer une borne ainsi qu'en ajouter/supprimer une, ainsi que changer son type. De plus, elle intègre dynamiquement les actions qui se déroulent automatiquement une fois le programme lancé. Nous permettons à l'utilisateur d'imprimer la liste des événements à tout moment (dans un fichier texte).

Le trafic gère un flux de moto, voiture, et un flux de camion. Tout ces flux peuvent être gérés dans l'interface graphique, en direct. Ils sont définis par un nombre de véhicules générés chaque minute. Nous permettons à l'utilisateur d'ajouter à sa guise des véhicules défectueux, qui bloqueront automatiquement le trafic de la borne où il se présentera (pendant un temps défini).

Les alarmes ont été créées avec différents types pour pouvoir spécifier à l'utilisateur la cause du problème survenu lors de l'appui sur le bouton d'une borne.

2 Description de la solution

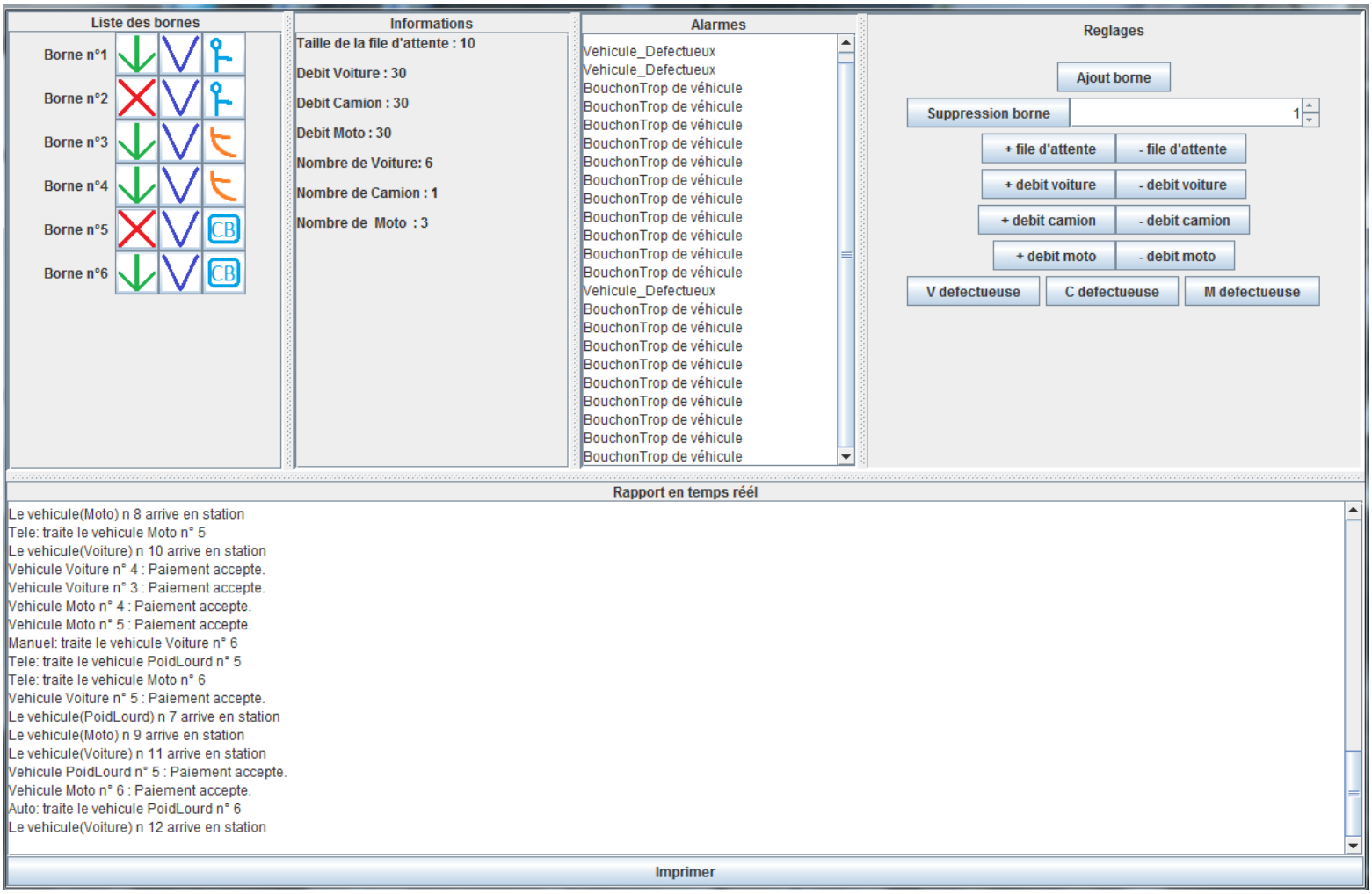
2.1 Exigences minimales

Le logiciel de simulation de barrière a pour but d'aider le client à pouvoir dimensionner son système réel avant sa réalisation effective.

- Paramétrer une simulation (nombre de véhicules, nombre de bornes).
- Imprimer un rapport (actuellement exporté sous la forme d'un fichier texte).
- Ajouter/Retirer dynamiquement une borne.
- Modifier le statut (ouvert/fermé) d'une borne existante, et son type.

2.2 Interface graphique (IHM)

Illustration 1: Fenêtre principale de l'IHM



3 Rôle des membres

3.1 Hiérarchie générale

Chef de projet : Guillaume MEUNIER

Interface graphique : Julien BRUYERE, Rodolphe ADAM

Gestion des véhicules : Thomas LOUIS, Alex PIRES

Gestion des bornes : Rémi GAZQUEZ, Nikola SLAVNIC

A noter qu'il y a eu une forte liaison entre les différents sous-groupes avec parfois une inversion des rôles, lorsqu'un problème avait été rencontré.

3.2 Descriptif détaillé des rôles respectifs

Guillaume :

Dans un premier temps, l'organisation du projet été un peu brouillon. Nous avions décidé de travailler sur Git avec le dépôt distant Github, et ayant le plus de connaissances dans ce domaine, je fut chargé de créer le repository. Nous avons commencer à coder sans vraiment savoir où aller suite à une modélisation non valide.

Je me suis alors proposé en tant que chef de ce projet pour le mener à bien. J'ai commencé à élaborer un tutoriel (présent dans le repository Git) pour expliquer l'installation du proxy, et de l'intégration de Git, sous Eclipse. Par la suite, j'ai élaboré ce que l'on peut appeler le cœur du système, à travers un complexe de liste de thread cherchant des objets dans une file d'attente, et d'une fabrique d'objet.

A partir de ce moment là, nous avons progressé par étape d'intégration. J'ai séparé le travail en binômes, et chaque binômes devait progresser un peu dans leur domaine, et ensuite me donner quelque chose de fonctionnel. Ainsi on testait à chaque petite étape pour voir si le logiciel marchait comme prévu. J'ai tenté tout le long de répartir le travail, et de faire en sorte que chaque personne produise quelque chose de fonctionnel en leur expliquant l'évolution de la modélisation.

La bonne communication au sein de l'équipe a permit d'avoir une intégration facile, beaucoup de challenges ont été relevé au cours du projet, et nous avons pu ainsi le mener à bien.

Julien et Rodolphe :

Rodolphe s'est chargé dans un premier temps de la mise en place de l'organisation globale du projet à travers le diagramme de Gantt. Celui-ci n'a pas été suivit à la lettre, mais nous a tout de même permis de nous fixer des délais plus fin que les échéances du projet demandées par les enseignants.

Julien Bruyère et Rodolphe Adam ont pris en charge la partie graphique du projet. Rodolphe s'est occupé en particulier de créer la fenêtre de réglages initiaux qui s'ouvre au lancement de l'application. Puis il s'est occupé d'une partie de la gestion en direct des paramètres de simulation et

l'affichage des informations sur la fenêtre principale. Julien a en parallèle créé l'ossature de la fenêtre principale qui se décompose en plusieurs interfaces (péage, informations, alarmes, réglages et rapport). Julien a ensuite géré toute l'interface péage et créé l'ossature des autres interfaces, ce qui a permis à Rodolphe de les remplir avec les éléments adéquats.

N'ayant jamais vraiment fait d'interface de logiciel, ce travail n'a pas été évident, d'autant plus que notre binôme se retrouvait donc en permanence dépendant du travail des autres binômes, puisque la modification des méthodes ajoutait ou supprimait des réglages.

Rémi :

En utilisant la première modélisation du diagramme des classes des bornes, avec Nikola on s'est occupés de toute la partie borne en respectant les demandes de thread et de fabrique proposé par Guillaume, ensuite on a tout regroupé sous la forme d'une BarrierPeage qui été la seule classe visible à l'équipe qui développait l'IHM.

J'ai ensuite aider l'équipe d'Alex pour concevoir dans la même optique une classe trafic qui regroupait les fabriques de véhicules.

Finalement j'ai rajouté à la demande de Guillaume plusieurs fonctionnalités comme le fait de pouvoir changer une borne pendant la simulation ou modifier le débit des fabriques de véhicules et j'ai aidé Julien et Rodolphe pour intégrer et corriger les problèmes qu'ils avaient avec l'IHM (relatifs au code que je leur avait mis a disposition). Pour ma part, je pense que 2 semaines d'analyse du sujet est trop court faire un diagramme des classes car il faut sans cesse revoir ses modélisations. De plus l'intégration est une partie extrêmement délicate, surtout quand on doit s'accorder avec une équipe qui développe une IHM pour la première fois.

Nikola :

Lors des premières séances nous nous sommes divisé le travail pour le cahier des charge et je me suis occupé du diagramme des classes avec Julien et Guillaume.

Ensuite, avec Rémi nous avons réfléchis et essayé de coder une première approche des bornes ainsi que des alarmes. Nous avons été amenés à modifier cette approche plusieurs fois en cours de route.

Je me suis aussi un peu occupé du code de la file d'attente des voitures.

Et pour finir j'ai décidé de refaire le diagramme de classe final.

Thomas et Alex:

Une fois l'analyse du sujet et la construction du diagramme de classe fait par le groupe, Je me suis occupé de la création de la classe véhicule et de ses classes dérivées. J'ai conçu avec Alex l'architecture du package véhicule. Une fois fait, il a fallu intégrer ce travail pour créer le prototype.

Thomas a ensuite écrit le système de Rapport puis de RapportPaiement et je l'ai intégré au travail commun en utilisant et en modifiant l'interface Rapport déjà créé par l'autre binôme de l'équipe. Après les tests passés sur ce qui avait été fait, nous avons conçu le système de Gestionnaire

d'Alarme pour en implémenter la détection des cas particulier qui a été associé au code de l'interface graphique associée.

Pour finir, nous avons aidé à intégrer mon travail dans le système global avec succès.

Alex s'est chargé de la reprise du cahier des charges pour produire le présent rapport final, avec l'aide de chacun des membres de l'équipe.

3.3 Diagramme de Gantt

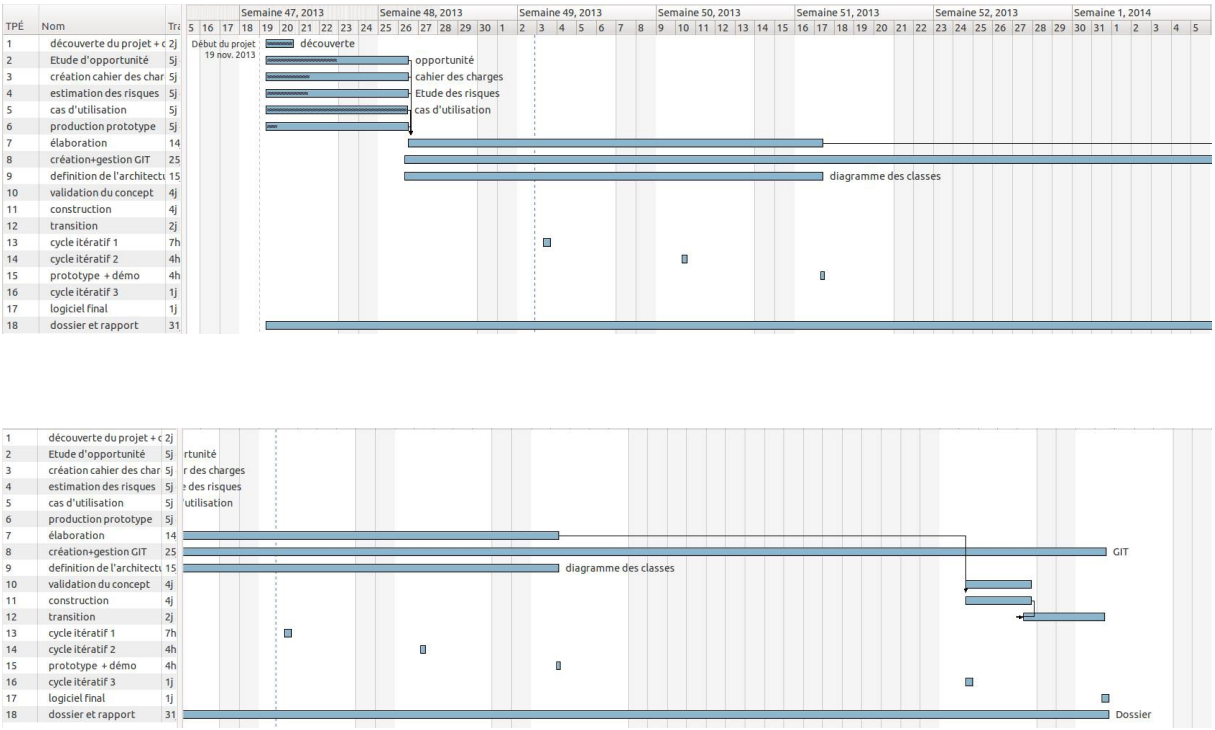


Illustration 2: Diagramme de Gantt

3 Outils de développement

3.1 Moyens de communication (données/idées)

- Groupe Facebook nous permet la communication jointe et une diffusion à tous les membres du groupe immédiate, organisations, centralisation de l'avancé de chacun.
- Google Drive permet le partage de documents avec une consultation en ligne possible pour un gain de temps (grâce à la gestion de nombreuses extensions courantes).
- Git centralise l'ensemble des fichiers de notre projet en permettant d'avoir en permanence une version accessible du projet pour tous les membres du groupe. Chaque implémentation est ajoutée par son réalisateur. Permet une gestion des conflits lors des nouveaux ajouts de modules modifié par plusieurs personne.

3.2 Outils de travail utilisés

- La modélisation (UML, DCU) a été réalisée à l'aide du logiciel ArgoUML.
- Pour toute la programmation du projet, nous avons utilisé l'IDE Eclipse. Installé sur les ordinateur de l'école, il est très ergonomique et nous l'utilisons déjà depuis la 1ère année pour développer en Java. Plusieurs extensions ont été nécessaires notamment Jigloo pour l'interface graphique du programme, ainsi que celle qui permet de récupérer les versions du dépôt Git.