Projet Développement Objet

Hexanôme 4203
Adenot Paul, Brodu Etienne, Gaudin Maxime
Golumbeanu Monica, Richard Martin, Rodière Yoann

24 novembre 2010

Table des matières

Première partie

Liste des besoins

0.1 Besoins communs

Besoin 1	Visualiser un objet dynamique
Besoin 2	Visualiser un objet statique
Besoin 3	Effectuer des actions sur l'historique

0.2 Configuration

Besoin 1	Effectuer des opérations sur un élément
Besoin 2	Simuler la configuration en cours
Besoin 3	Gérer la persistance d'une configuration

0.3 Simulation

Besoin 1	Charger une configuration pour la simulation
Besoin 2	Gérer la liste des vols
Besoin 3	Changer le mode de simulation entre automatique et manuel
Besoin 4	Agir manuellement sur les éléments visualisés
Besoin 5	Effectuer des opérations sur l'avancement de la simulation
Besoin 6	Gérer les événements
Besoin 7	Gérer la persistance d'une simulation
Besoin 8	Mettre à jour l'état du système ponctuellement l'état du système
Besoin 9	Visualiser un objet dynamique
Besoin 10	Mettre à jour périodiquement l'état du système

0.4 Exploitation

Besoin 1	Charger une configuration
Besoin 2	Gérer la persistance des fichiers de journalisation
Besoin 3	Effectuer un arrêt d'urgence
Besoin 4	Effectuer des opérations protégées sur des éléments
Besoin 5	Acheminer automatiquement les bagages
Besoin 6	Mettre à jour l'état du système périodiquement
Besoin 7	Mettre à jour l'état du système ponctuellement
Besoin 8	Visualiser un objet dynamique

0.5 Maintenance

Besoin 1	Effectuer des opérations non protégées sur un élément
Besoin 2	Visualiser les résultats/problèmes
Besoin 3	Gérer les interventions

0.6 Réclamation

Besoin 1	Gérer un dossier de litige
Besoin 2	Se renseigner sur un bagage
Besoin 3	Gérer la persistance des dossiers de litiges

Deuxième partie
Cas d'utilisation

1 Rôles

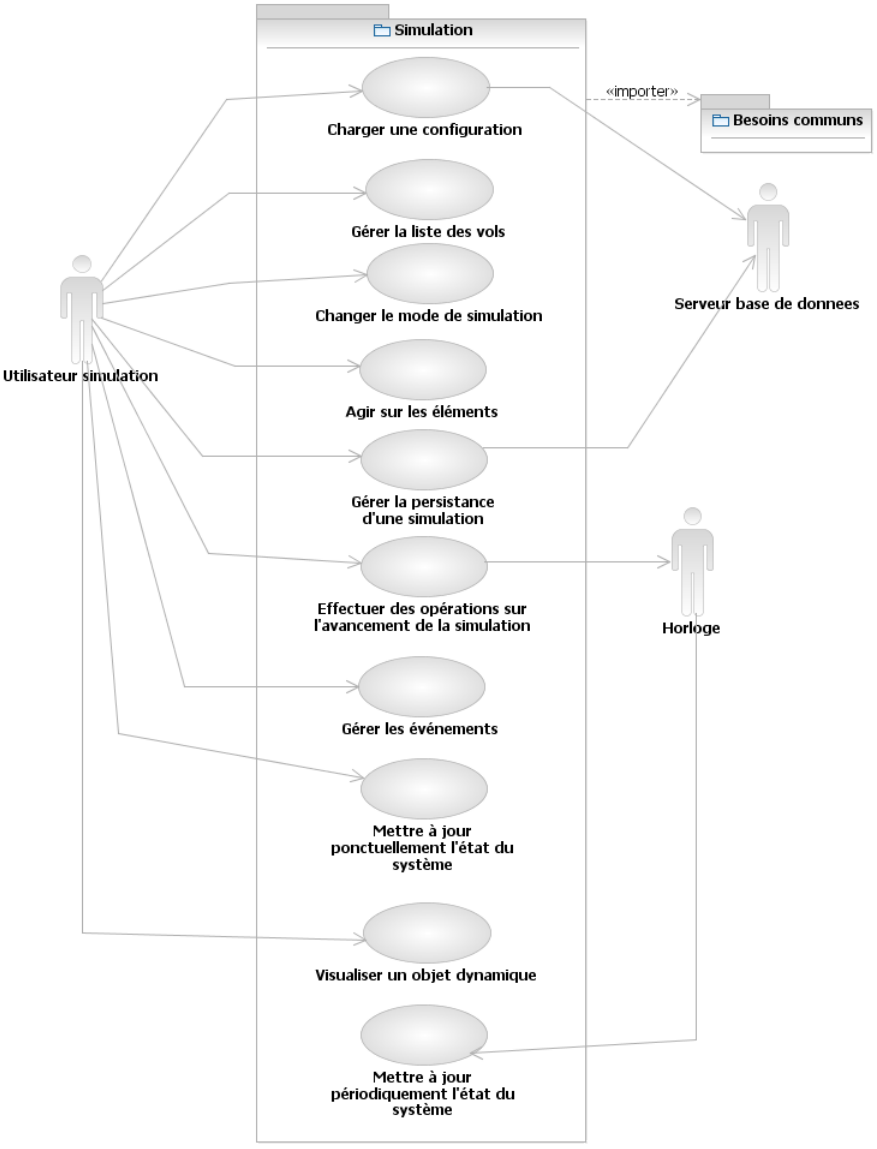

FIGURE 1 – *Hiérarchie des rôles*

Utilisateur client graphique — C'est une personne ayant accès aux fonctionnalités graphiques du logiciel SGBag. Les autres rôles « Utilisateurs » en dérivent.

Utilisateur configuration — C'est une personne qui manipule l'interface de configuration. Seul le responsable technique possède les autorisations pour modifier ou créer des configurations.

Utilisateur simulation — C'est une personne qui manipule l'interface de simulation. Le responsable technique et le superviseur en font partie.

Utilisateur exploitation	C'est une personne qui manipule l'interface de maintenance. Le superviseur et l'informaticien sont les deux employés pouvant accéder à cette interface.
Utilisateur maintenance	C'est une personne qui manipule l'interface de maintenance. Le superviseur et l'informaticien sont les deux employés pouvant accéder à cette interface.
Utilisateur gestion réclamation	C'est une personne qui gère les problèmes survenus en cas de perte ou de dégradation des bagages. Elle peut accéder au dossier des litiges ainsi qu'au trajet des bagages.
Objet dynamique	Objet commandable en mouvement, à distance, tel qu'un tapis, un chariot, ...
Capteur actif	Capteur interrompant le système lorsqu'il doit transmettre une information (alarme, ...).
Capteur passif	Capteur dont la valeur doit être lue à intervalle régulier (caméra, ...).
Horloge	Une horloge, déclenchant des événement à une fréquence donnée.
Serveur base de données	La base de donnée du système de gestion de bagages contient toutes les données de configuration et de gestion de l'application.
PDA Technicien	C'est le PDA [a] de la personne qui intervient en cas de problème.

a. Personal Digital Assistant

2 Cas d'utilisation

2.1 Besoins communs

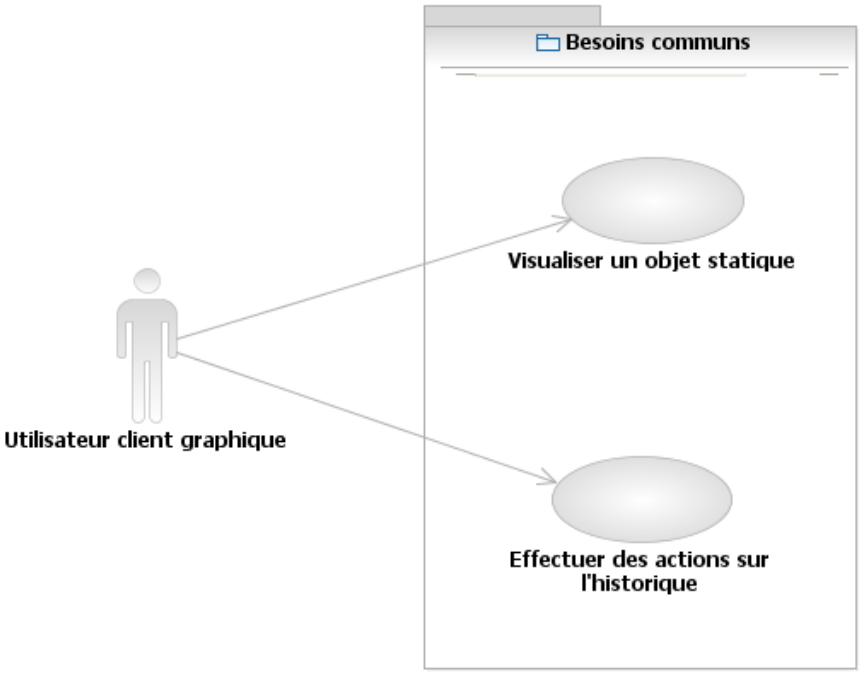

FIGURE 2 – *Cas d'utilisations : Besoins communs*

2.2 Visualiser un objet statique

L'utilisateur du client graphique (1) ou *L'utilisateur de l'interface de configuration* (1) peut sélectionner, zoomer ou dézoomer sur un objet de haut niveau.

2.3 Effectuer des actions sur l'historique

L'utilisateur du client graphique (1) peut annuler les actions qu'il vient de réaliser ou à l'inverse les reproduire après les avoir annulé.

2.4 Configuration

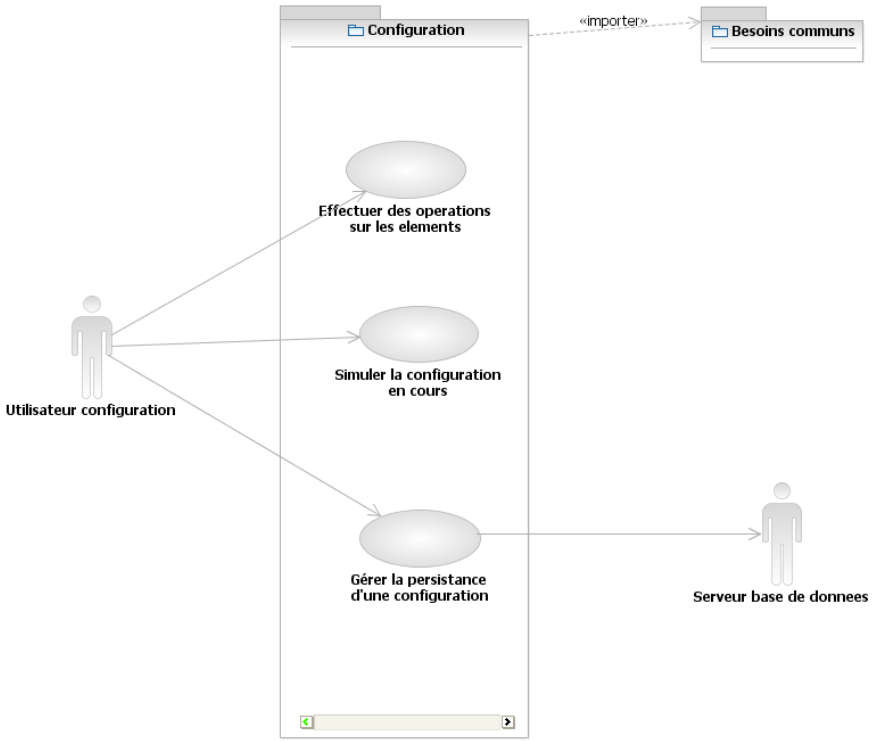

FIGURE 3 – *Cas d'utilisations : Configuration*

2.4.1 Effectuer des opérations sur un élément

L'utilisateur de la configuration (1) peut visualiser, ajouter, déplacer, lier aux autres éléments, paramétrer ou supprimer un élément de la configuration courante.

2.4.2 Simuler la configuration en cours

L'utilisateur de la configuration (1) peut ouvrir l'interface de simulation à partir de l'interface de configuration. Les paramètres de la simulation sont ceux de la configuration en cours.

2.4.3 Gérer la persistance d'une configuration

L'utilisateur de la configuration (1) peut enregistrer, enregistrer sous, charger, créer, supprimer ou dupliquer la configuration dans la *base de donnée* (1) .

2.5 Simulation

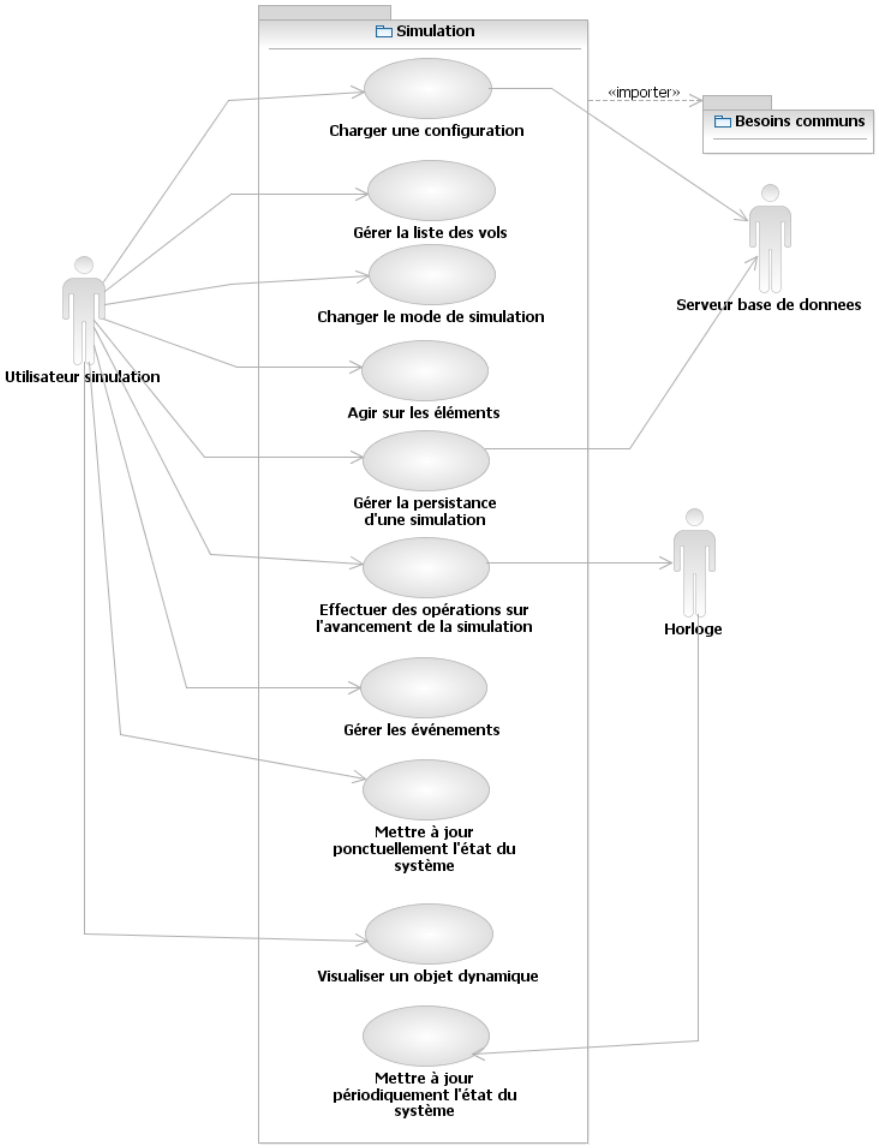

FIGURE 4 – *Cas d'utilisations : Simulation*

2.5.1 Charger une configuration

L'utilisateur de la simulation (1) peut définir la configuration d'aéroport utilisée pour la simulation, à partir de la *base de donnée* (1) .

2.5.2 Gérer la liste des vols

L'utilisateur de la simulation (1) peut ajouter, retirer, paramétrer des vols à simuler.

2.5.3 Changer le mode de simulation

L'utilisateur de la simulation (1) peut basculer entre mode manuel et automatique.

2.5.4 Agir manuellement sur les éléments

L'utilisateur de la simulation (1) peut sélectionner, paramétrer, mettre en marche/arrêt un objet dynamique.

2.5.5 Effectuer des opérations sur l'avancement de la simulation

L'utilisateur de la simulation (1) peut démarrer, stopper, mettre en pause, modifier la vitesse de la simulation et par conséquent modifier les paramètres de fonctionnement de *l'horloge* (1) .

2.5.6 Gérer les événements

L'utilisateur de la simulation (1) Créer, modifier, supprimer, visualiser, activer/désactiver des événements.

2.6 Gérer la persistance d'une simulation

L'utilisateur de la simulation (1) peut enregistrer, enregistrer sous, charger, créer, supprimer ou dupliquer la configuration de la simulation dans la *base de donnée* (1)

2.7 Mettre à jour l'état du système ponctuellement

L'utilisateur de la simulation (1) peut mettre à jour l'état du système, *i.e.* les positions, l'état des différents objets, redessiner *etc.*

2.8 Mettre à jour l'état du système périodiquement

L'horloge (1) peut mettre à jour l'état du système.

2.9 Visualiser un objet dynamique

L'utilisateur de l'interface de simulation (1) peut sélectionner, zoomer ou dézoomer sur un chariot ou un avion.

2.10 Exploitation

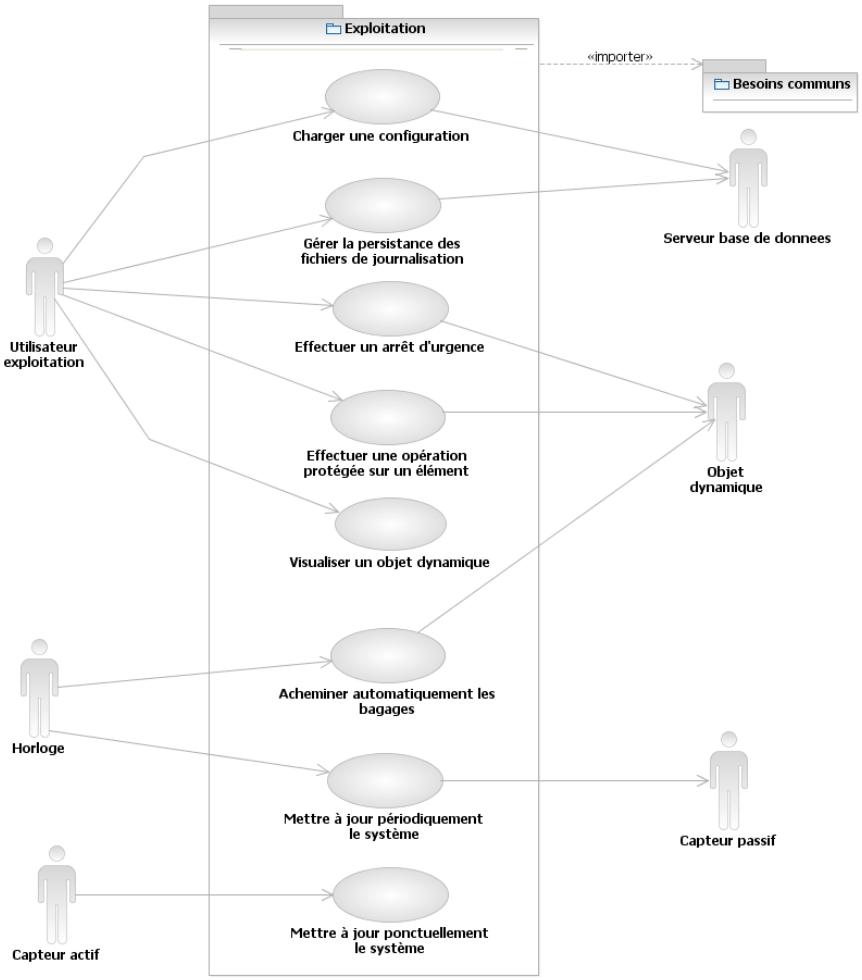

FIGURE 5 – *Cas d'utilisations : Exploitation*

2.10.1 Charger une configuration

L'utilisateur exploitation (1) peut définir la configuration d'aéroport utilisée pour l'exploitation, à partir de la *base de donnée* (1) .

2.11 Gérer la persistance des fichiers de journalisation

L'utilisateur exploitation (1) peut enregistrer, enregistrer sous, charger, crée, supprimer ou dupliquer un fichier de journalisation à partir, ou dans, la *base de donnée* (1) .

2.12 Effectuer un arrêt d'urgence

L'utilisateur exploitation (1) peut effectuer un arrêt d'urgence. Commande l'arrêt d'urgence de tous les *objets dynamiques* (1) .

2.12.1 Effectuer une opération protégée sur un éléments

L'utilisateur exploitation (1) peut paramétrer, arrêter ou démarrer *l'objet dynamique* (1) .

2.12.2 Acheminer automatiquement les bagages

À chaque ticks d' *horloge* (1) , et lorsque c'est nécessaire, le système détermine le chemin que chaque *objet dynamique* (1) doit emprunter.

2.13 Mettre à jour l'état du système périodiquement

L'horloge (1) peut mettre à jour l'état du système en consultant les *capteurs passifs* (1) .

2.14 Mettre à jour l'état du système ponctuellement

Un capteur actif (1) peut mettre à jour l'état du système.

2.15 Visualiser un objet dynamique

L'utilisateur exploitation (1) peut sélectionner, zoomer ou dézoomer sur un chariot ou un avion.

2.16 Maintenance

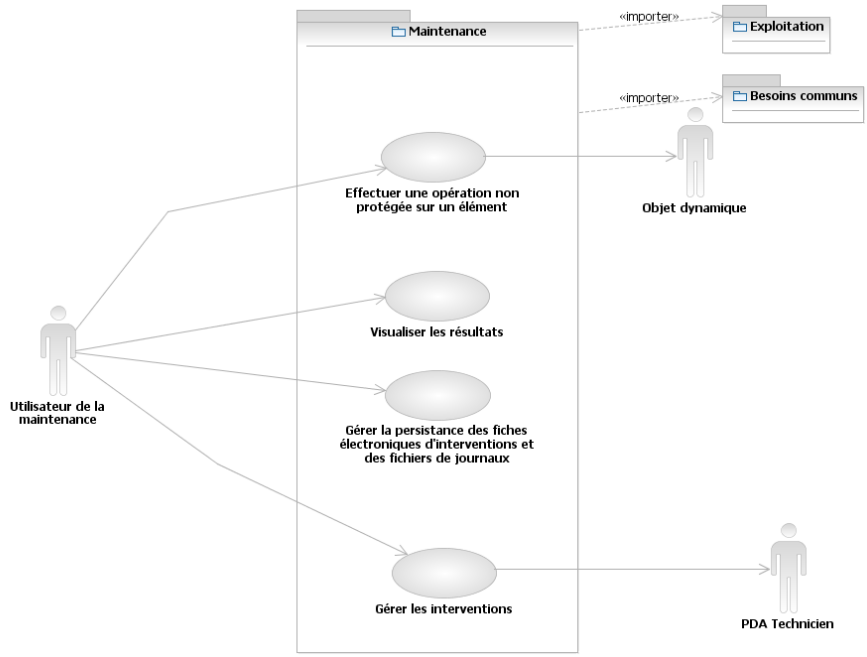

FIGURE 6 – *Cas d'utilisations : Maintenance*

2.16.1 Effectuer des opérations non protégées sur un élément

L'utilisateur de la maintenance (1) peut effectuer des opérations habituellement interdites par le système (*e.g.* ajouter dans un conteneur déjà surchargé) sur un *objet dynamique* (1) .

2.16.2 Visualiser les résultats

L'utilisateur de la maintenance (1) peut visualiser le comportement du système (via une constatation sur le terrain) à la suite d'opération d'exploitation sur le système.

2.16.3 Gérer la persistance des fiches électroniques d'interventions et des fichiers de journaux

L'utilisateur de la maintenance (1) peut créer, modifier, supprimer, ou renommer un fichier journal ou une fiche d'intervention de la *base de donnée* (1) .

2.16.4 Gérer les interventions

L'utilisateur de la maintenance (1) peut, en cas de problème, assigner la résolution d'un problème à un technicien via *son PDA technicien* (1) et créer un fichier électronique qui contient un rapport d'intervention en *base de donnée* (1) .

2.17 Réclamation

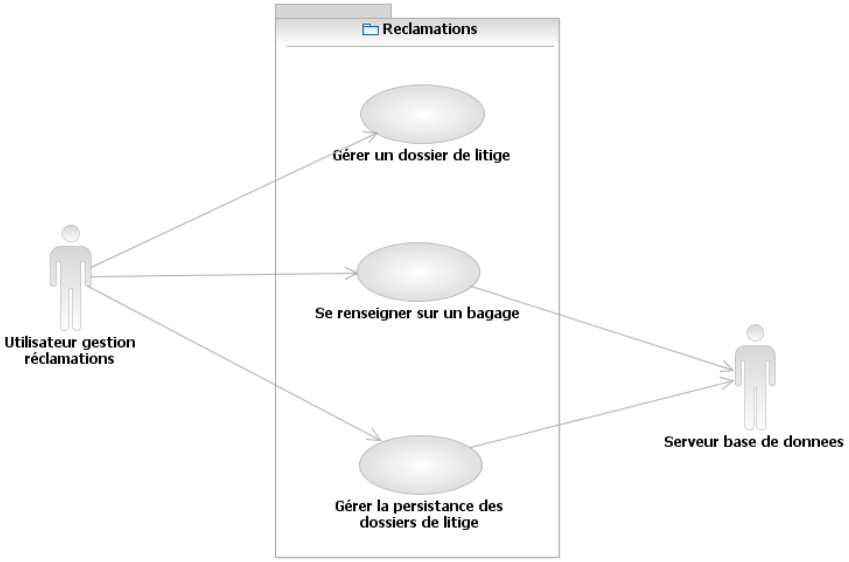

FIGURE 7 – *Cas d'utilisations : Réclamation*

2.17.1 Gérer un dossier de litige

En cas de litige, *l'utilisateur gestion réclamations* (1) peut (perte ou dégradation d'un bagage), ouvrir, modifier, visualiser ou fermer un dossier de litige. Il peut aussi créer un nouveau couple identifiant/Mot de passe voyageur.

2.17.2 Se renseigner sur un bagage

L'utilisateur gestion réclamation (1) peut visualiser le trajet d'un bagage pendant son traitement par le système de gestion à partir des données de *la base de donnée* (1) .

2.17.3 Gérer la persistance des dossiers de litiges

L'utilisateur gestion réclamation (1) peut charger, sauvegarder ou supprime un fichier journal ou une fiche d'intervention de la *base de donnée* (1) .

Troisième partie
Scénarios

*Les scenarios précédés d'un * seront dans le prototype.*

3 Besoins communs

3.1 Visualisation statique

* **Scénario 1** Selectionner l'objet statique
Stimulis Utilisateur client graphique

* **Scénario 2** Zoomer l'objet statique
Stimulis Utilisateur client graphique

* **Scénario 3** Dé-zoomer l'objet statique
Stimulis Utilisateur client graphique

3.2 Visualisation dynamique

* **Scénario 1** Selectionner l'objet dynamique
Stimulis Utilisateur client graphique

* **Scénario 2** Zoomer l'objet dynamique
Stimulis Utilisateur client graphique

* **Scénario 3** Dé-zoomer l'objet dynamique
Stimulis Utilisateur client graphique

4 Configuration

4.1 Effectuer des opérations sur un élément

Scénario 1 Ajouter un élément
Stimulis Utilisateur configuration

Scénario 2 Visualiser un élément
Stimulis Utilisateur configuration

Scénario 3 Déplacer un élément
Stimulis Utilisateur configuration

Scénario 4 Lier aux autres éléments
Stimulis Utilisateur configuration

Scénario 5 Supprimer un élément
Stimulis Utilisateur configuration

Scénario 6 Paramètrer un élément
Stimulis Utilisateur configuration

4.2 Simuler

Scénario 1 Simuler
Stimulis Utilisateur configuration

4.3 Gérer la persistance de la configuration

Scénario 1 Enregistrer une configuration
Stimulis Utilisateur configuration

Scénario 2 Enregistrer-sous une configuration
Stimulis Utilisateur configuration

Scénario 3 Charger une configuration
Stimulis Utilisateur configuration

Scénario 4 Modifier une configuration
Stimulis Utilisateur configuration

Scénario 5 Ouvrir une configuration
Stimulis Utilisateur configuration

Scénario 6 Supprimer une configuration
Stimulis Utilisateur configuration

Scénario 7 Dupliquer une configuration
Stimulis Utilisateur configuration

5 Simulation

5.1 Charger une configuration pour la simulation

Scénario 1 Charger une simulation
Stimulis Utilisateur simulation

Scénario 2 Charger une configuration pour la simulation
Stimulis Utilisateur simulation

5.2 Gérer la persistance d'une simulation

Scénario 1 Enregistrer une simulation
Stimulis Utilisateur simulation

Scénario 2 Enregistrer-sous une simulation
Stimulis Utilisateur simulation

Scénario 3 Créer une simulation
Stimulis Utilisateur simulation

Scénario 4 Supprimer une simulation
Stimulis Utilisateur simulation

Scénario 5 Dupliquer une simulation
Stimulis Utilisateur simulation

Scénario 6 Modifier une simulation
Stimulis Utilisateur simulation

5.3 Gérer la liste des vols

* **Scénario 1** Créer un vol
Stimulis Utilisateur simulation

* **Scénario 2** Modifier un vol
  Stimulis Utilisateur simulation
* **Scénario 3** Supprimer un vol
  Stimulis Utilisateur simulation

5.4 Changer le mode de simulation

* **Scénario 1** Passer en mode automatique
  Stimulis Utilisateur simulation
* **Scénario 2** Passer en mode manuel
  Stimulis Utilisateur simulation

5.5 Agir sur les éléments visualisés

* **Scénario 1** Sélectionner un élément
  Stimulis Utilisateur simulation
* **Scénario 2** Paramétrer un élément
  Stimulis Utilisateur simulation
* **Scénario 3** Mettre en marche un élément
  Stimulis Utilisateur simulation
* **Scénario 4** Arrêter un élément
  Stimulis Utilisateur simulation

5.6 Effectuer des opérations sur l'avancement de la simulation

* **Scénario 1** Démarrer une simulation
  Stimulis Utilisateur simulation
* **Scénario 2** Stopper une simulation
  Stimulis Utilisateur simulation
* **Scénario 3** Mettre en pause une simulation
  Stimulis Utilisateur simulation
* **Scénario 4** Changer la vitesse de simulation
  Stimulis Utilisateur simulation

5.7 Gérer les événements

* **Scénario 1** Créer un événement
  Stimulis Utilisateur simulation
* **Scénario 2** Modifier un événement
  Stimulis Utilisateur simulation
* **Scénario 3** Supprimer un événement
  Stimulis Utilisateur simulation
* **Scénario 4** Visualiser un événement
  Stimulis Utilisateur simulation
* **Scénario 5** Activer un événement
  Stimulis Utilisateur simulation

* **Scénario 6** Désactiver un événement
  Stimulis Utilisateur simulation

5.8 Déclencher les événements

* **Scénario 1** Déclencher un événement
  Stimulis Horloge

5.9 Mettre à jour l'état du système

* **Scénario 1** Prendre en compte l'interruption d'un capteur actif
  Stimulis Capteur actif
* **Scénario 2** Prendre en compte les valeurs des capteurs passifs
  Stimulis Horloge

6 Exploitation

6.1 Effectuer une opération protégée

Scénario 1 Paramétrer un élément de manière sécurisée
Stimulis Utilisateur exploitation

Scénario 2 Arrêter un élément de manière sécurisée
Stimulis Utilisateur exploitation

Scénario 3 Démarrer un élément de manière sécurisée
Stimulis Utilisateur exploitation

6.2 Acheminer automatiquement les bagages

Scénario 1 Acheminer automatiquement les bagages
Stimulis Horloge

6.3 Arrêter d'urgence le système

Scénario 1 Arrêter le système
Stimulis Utilisateur exploitation

6.4 Gérer la persistance des configurations et des fichiers de logs

Scénario 1 Charger une configuration
Stimulis Utilisateur exploitation

6.5 Mettre à jour l'état du système

Scénario 1 Prendre en compte l'interruption d'un capteur actif
Stimulis Capteur actif

Scénario 2 Prendre en compte les valeurs des capteurs passifs
Stimulis Horloge

6.6 Maintenance

6.7 Visualiser les résultats

Scénario 1 Visualiser le comportement du système
Stimulis Utilisateur maintenance

6.8 Gérer les interventions

Scénario 1 Assigner une intervention à un technicien
Stimulis Utilisateur maintenance

Scénario 2 Créer un fichier électronique avec le rapport d'intervention
Stimulis Utilisateur maintenance

6.9 Gérer la persistance des fichiers électroniques d'intervention et des fichiers de journaux

Scénario 1 Enregistrer un fichier électronique d'intervention
Stimulis Utilisateur maintenance

Scénario 2 Lire un fichier électronique d'intervention
Stimulis Utilisateur maintenance

Scénario 3 Supprimer un fichier électronique d'intervention
Stimulis Utilisateur maintenance

Scénario 4 Modifier un fichier électronique d'intervention
Stimulis Utilisateur maintenance

Scénario 5 Lire un fichier de journal
Stimulis Utilisateur maintenance

Scénario 6 Supprimer un fichier de journal
Stimulis Utilisateur maintenance

6.10 Réclamation

6.11 Gérer un dossier de litige

Scénario 1 Créer un dossier de lititge
Stimulis Utilisateur réclamation

Scénario 2 Modifier un dossier de lititge
Stimulis Utilisateur réclamation

Scénario 3 Visualiser un dossier de lititge
Stimulis Utilisateur réclamation

Scénario 4 Fermer un dossier de lititge
Stimulis Utilisateur réclamation

6.12 Créer un nouvel identifiant voyageur

Scénario 1 Créer un numéro d'identification et un mot de passe associé à un voayageur lésé
Stimulis Utilisateur réclamation

6.13 Se renseigner sur un bagage

Scénario 1 Visualiser le trajet d'un bagage
Stimulis Utilisateur réclamation

6.14 Gérer la persistance des dossiers de litiges

Scénario 1 Enregistrer un dossier de litige
Stimulis Utilisateur réclamation

Scénario 2 Charger un dossier de litige
Stimulis Utilisateur réclamation

Scénario 3 Supprimer un dossier de litige
Stimulis Utilisateur réclamation

Quatrième partie

Listings

fenetreprincipale.cpp main.cpp vueelement.cpp vuetapis.cpp vuetroncon.cpp vuevol.cpp Bagage.cpp Chariot.cpp Noeud.cpp Prototype.cpp StrategiePilotage.cpp StrategiePilotageManuel.cpp Tapis.cpp XmlConfigFactory.cpp

```
#include <QGraphicsSvgItem>
#include <QtSvg/QSvgRenderer>

#include "src/ihm/vuebagage.h"
#include "src/ihm/vueconfig.h"

using namespace vue_config::bagage;

QSvgRenderer *VueBagage::_renderer = new QSvgRenderer(bagageSimple);

VueBagage::VueBagage(FenetrePrincipale& fenetrePrincipale, Bagage &bagage):
        VueElement(fenetrePrincipale,rect),
        _image(new QGraphicsSvgItem()),
        _bagage(bagage)
{
    _image->setSharedRenderer(_renderer);

    setCacheMode(QGraphicsItem::DeviceCoordinateCache);
    _image->setCacheMode(QGraphicsItem::DeviceCoordinateCache);

    setZValue(zIndex);
}


void VueBagage::advance(int pas)
{
    if(pas == 0)
    {
            return;
    }
    setPos(_bagage.position());
}

void VueBagage::paint(QPainter *painter, const QStyleOptionGraphicsItem *,
    QWidget *)
{
    VueElement::paint(painter, 0, 0);

    _image->renderer()->render(painter, boundingRect());
}

Bagage* VueBagage::bagageAssocie()
{
    return &_bagage;
}
```

```
#include <QLineF>

#include "vueconfig.h"
```

```
#include "vuecanevas.h"
#include "fenetreprincipale.h"

using namespace vue_config::canevas;

VueCanevas::VueCanevas(FenetrePrincipale& fenetrePrincipale) :
    _fenetrePrincipale(fenetrePrincipale)
{
    setFlags(QGraphicsItem::ItemIsSelectable);
}

VueCanevas::VueCanevas(FenetrePrincipale& fenetrePrincipale,
                       QRectF rect) :
    _fenetrePrincipale(fenetrePrincipale),
    _rect(rect)
{
    setFlags(QGraphicsItem::ItemIsSelectable);
}

void VueCanevas::definirCoordonnees(QPointF positionDebut,
                                QPointF positionFin, double largeur,
                                double ajoutLongueurApresFin,
                                double ajoutLongueurAvantDebut)
{
    QLineF direction(positionDebut, positionFin);
    direction.setLength(direction.length()
                        + ajoutLongueurApresFin + ajoutLongueurAvantDebut);
    direction.translate(QVector2D(positionDebut-positionFin).normalized()
                        .toPointF()*ajoutLongueurAvantDebut);

    _rect = QRectF(0,-largeur/2,direction.length(), largeur);

    setPos(direction.p1());
    setRotation(-direction.angle());
}

QRectF VueCanevas::boundingRect() const
{
    QRectF rect = _rect;
    rect.setTopLeft(rect.topLeft() - QPoint(marge,marge));
    rect.setBottomRight(rect.bottomRight() + QPoint(marge,marge));

    return rect;
}
```

```
#include "src/ihm/vuechariot.h"
#include "src/ihm/vueconfig.h"

#include <QGraphicsSvgItem>
#include <QtSvg/QSvgRenderer>

#include <sys/time.h>
#include <unistd.h>

using namespace vue_config::chariot;

QSvgRenderer *VueChariot::_renderer = new QSvgRenderer(etatNormal);

VueChariot::VueChariot(FenetrePrincipale& fenetrePrincipale, Chariot &
    chariot):
```

```
15          VueElement(fenetrePrincipale,rect),
16          _image(new QGraphicsSvgItem()),
17          _chariot(chariot)
18  {
19      _image->setSharedRenderer(_renderer);
20
21      setCacheMode(QGraphicsItem::DeviceCoordinateCache);
22      _image->setCacheMode(QGraphicsItem::DeviceCoordinateCache);
23
24      setZValue(zIndex);
25      setPos(chariot.position());
26  }
27
28  void VueChariot::advance(int pas)
29  {
30          if(pas == 0)
31          {
32                  return;
33          }
34          setPos(_chariot.position());
35  }
36
37  void VueChariot::paint(QPainter *painter, const QStyleOptionGraphicsItem *,
        QWidget *)
38  {
39      VueElement::paint(painter, 0, 0);
40
41          _image->renderer()->render(painter, rect);
42
43          /* Affichage de l'id du chariot
44             Valable avec les nouveaux rendus.
45
46          painter->setFont (font);
47          QTransform matriceActuelle = _paintPixmap.transform();
48          QTransform matriceTexte;
49          matriceTexte.translate(dxTexte,dyTexte);
50          matriceTexte.rotate(rotationTexte);
51          painter->setTransform(matriceTexte);
52          painter->setPen(couleurTexte);
53          painter->drawText(QRectF(0,0,200,100), Qt::AlignLeft, QString::
                number(_chariot.id()));
54          painter->setTransform(matriceActuelle);*/
55  }
56
57  Chariot& VueChariot::chariot()
58  {
59      return _chariot;
60  }
```

```
1   #include "vueparametreschariot.h"
2   #include "ui_vueparametreschariot.h"
3   #include "src/noyau/Chariot.h"
4
5   VueParametresChariot::VueParametresChariot(Chariot& chariot, QWidget *
        parent) :
6       QWidget(parent),
7       ui(new Ui::VueParametresChariot),
8       _chariot(chariot)
9   {
```

```
10      ui->setupUi(this);
11
12      connect(ui->vitesseMax,SIGNAL(valueChanged(double)),
13              &_chariot,SLOT(modifierVitesseMax(double)));
14      ui->vitesseMax->setValue(_chariot.vitesseMax());
15
16      connect(&chariot,SIGNAL(vitesseModifiee(double)),
17              ui->vitesseActuelle,SLOT(setValue(double)));
18      ui->vitesseActuelle->setValue(_chariot.vitesse());
19
20      connect(ui->modePilotage, SIGNAL(valueChanged(int)),
21              this,SLOT(modifierTypePilotage(int)));
22      ui->modePilotage->setValue(_chariot.typePilotage());
23
24      connect(ui->directionConseillee, SIGNAL(valueChanged(int)),
25              this,SLOT(modifierDirectionConseillee(int)));
26      ui->directionConseillee->setValue(_chariot.directionConseillee());
27
28      connect(ui->actif,SIGNAL(toggled(bool)),
29              &_chariot,SLOT(definirActivation(bool)));
30      connect(&_chariot,SIGNAL(activationModifiee(bool)),
31              ui->actif,SLOT(setChecked(bool)));
32      ui->actif->setChecked(_chariot.estActif());
33  }
34
35  VueParametresChariot::~VueParametresChariot()
36  {
37      delete ui;
38  }
39
40  void VueParametresChariot::modifierTypePilotage(int typePilotage)
41  {
42      _chariot.modifierTypePilotage(
43              static_cast<Chariot::TypePilotage>(typePilotage));
44  }
45
46  void VueParametresChariot::modifierDirectionConseillee(int direction)
47  {
48      _chariot.modifierDirectionConseillee(
49              static_cast<Direction>(direction));
50  }
```

```
1   #include "src/ihm/vueparametreselementactif.h"
2   #include "ui_vueparametreselementactif.h"
3
4   VueParametresElementActif::VueParametresElementActif(QWidget *parent) :
5       QWidget(parent),
6       ui(new Ui::VueParametresElementActif)
7   {
8       ui->setupUi(this);
9   }
10
11  VueParametresElementActif::~VueParametresElementActif()
12  {
13      delete ui;
14  }
```

```
1   #include "vueparametrestoboggan.h"
2   #include "ui_vueparametrestoboggan.h"
3
```

```
4  VueParametresToboggan::VueParametresToboggan(Toboggan& toboggan, QWidget *
       parent) :
5      QWidget(parent),
6      _toboggan(toboggan),
7      ui(new Ui::VueParametresToboggan)
8  {
9      ui->setupUi(this);
10     connect(&_toboggan, SIGNAL(nombreDeBagages(int)), this, SLOT(
           nombreDeBagages(int)));
11     ui->nbBagagesSpinBox->setValue(_toboggan.nombreDeBagages());
12 }
13
14 VueParametresToboggan::~VueParametresToboggan()
15 {
16     delete ui;
17 }
18
19 void VueParametresToboggan::nombreDeBagages(int nbBagages)
20 {
21     ui->nbBagagesSpinBox->setValue(nbBagages);
22 }
```

```
1  #include "src/ihm/vuetoboggan.h"
2  #include "src/ihm/vueconfig.h"
3
4  #include "src/noyau/Toboggan.h"
5
6  #include <QtSvg/QSvgRenderer>
7  #include <QVector2D>
8
9  using namespace vue_config::toboggan;
10
11 QSvgRenderer *VueToboggan::_renderer = new QSvgRenderer(etatNormal);
12
13 VueToboggan::VueToboggan(FenetrePrincipale& fenetrePrincipale, Toboggan &
       toboggan):
14         VueElement(fenetrePrincipale),
15         _image(new QGraphicsSvgItem()),
16         _toboggan(toboggan)
17 {
18     _image->setSharedRenderer(_renderer);
19
20     setCacheMode(QGraphicsItem::DeviceCoordinateCache);
21     _image->setCacheMode(QGraphicsItem::DeviceCoordinateCache);
22
23     setZValue(zIndex);
24     definirCoordonnees(_toboggan.position(),_toboggan.pointConnexion(),
25                        largeur,-vue_config::chariot::largeur/2);
26
27     _image->renderer()->setFramesPerSecond(30);
28 }
29
30 void VueToboggan::associerVol(Vol* vol)
31 {
32     _toboggan.associerVol(vol);
33 #ifdef DEBUG_ACHEMINEMENT
34     qDebug() << _toboggan;
35 #endif
36     vol->associer(&_toboggan);
```

```
}



void VueToboggan::advance(int pas)
{
        if(pas == 0)
        {
                return;
        }
}

void VueToboggan::paint(QPainter *painter, const QStyleOptionGraphicsItem
    *, QWidget *)
{
    VueElement::paint(painter, 0, 0);
    _image->renderer()->render(painter, boundingRect());
}

Toboggan& VueToboggan::toboggan() const
{
    return _toboggan;
}
```

```
#ifdef DEBUG_ACHEMINEMENT
#include <QDebug>
#include <QString>
#endif

#include "Element.h"
#include "XmlConfigFactory.h"

//Begin section for file Element.cpp
//TODO: Add definitions that you want preserved
//End section for file Element.cpp


//@uml.annotationsderived_abstraction="platform:/resource/usdp/
    ModeleStructurel.emx#_Y1tCkOsVEd-oy8D834IawQ?DEFCONSTRUCTOR"
//@generated "UML to C++ (com.ibm.xtools.transform.uml2.cpp.
    CPPTransformation)"
Element::Element(const XmlConfigFactory::IndexParamValeur& indexParamValeur
    ) :
        _position(indexParamValeur[XmlConfigFactory::NodeName_String[
            XmlConfigFactory::x]].toFloat(),
                indexParamValeur[XmlConfigFactory::NodeName_String[
                    XmlConfigFactory::y]].toFloat())
#ifdef DEBUG_ACHEMINEMENT
        ,
        _id (indexParamValeur[XmlConfigFactory::NodeName_String[
            XmlConfigFactory::id]].toInt()),
        _typeElement(indexParamValeur[XmlConfigFactory::NodeName_String[
            XmlConfigFactory::typeElement]])
#endif
{
```

```
25      // Vide
26  }
27
28  void Element::init (const XmlConfigFactory::IndexParamValeur& /*
        indexParamValeur*/,
29                      XmlConfigFactory& /*fabrique*/)
30  {
31      // Vide
32  }
33
34  //@uml.annotationsderived_abstraction="platform:/resource/usdp/
        ModeleStructurel.emx#_Y1tCkOsVEd-oy8D834IawQ?DESTRUCTOR"
35  //@generated "UML to C++ (com.ibm.xtools.transform.uml2.cpp.
        CPPTransformation)"
36  Element::~Element()
37  {
38      //TODO Auto-generated method stub
39  }
40
41  //@uml.annotationsderived_abstraction="platform:/resource/usdp/
        ModeleStructurel.emx#_h21_IPD4Ed-R6YEVT5cViQ"
42  //@generated "UML to C++ (com.ibm.xtools.transform.uml2.cpp.
        CPPTransformation)"
43  QPointF Element::position() const
44  {
45      return _position;
46  }
47
48  #ifdef DEBUG_ACHEMINEMENT
49  QDebug operator<<(QDebug dbg, const Element &element)
50  {
51      dbg.nospace() << element._typeElement.toAscii().data() << "(" << element
            ._id << ")";
52
53      return dbg.space();
54  }
55  #endif
56
57  int Element::id()
58  {
59      return _id;
60  }
```

```
1   #include "ElementActif.h"
2   //Begin section for file ElementActif.cpp
3   //TODO: Add definitions that you want preserved
4   //End section for file ElementActif.cpp
5
6   const double ElementActif::VITESSE_DEFAUT = 10; // m/s
7   const double ElementActif::VITESSE_MAX_DEFAUT = 20; // m/s
8
9   //@uml.annotationsderived_abstraction="platform:/resource/usdp/
        ModeleStructurel.emx#_Pu31cPGuEd-1y9a3HOSRUA?DEFCONSTRUCTOR"
10  //@generated "UML to C++ (com.ibm.xtools.transform.uml2.cpp.
        CPPTransformation)"
11  ElementActif::ElementActif(const XmlConfigFactory::IndexParamValeur&
        indexParamValeur) :
12          Element(indexParamValeur),
13          _vitesse (VITESSE_DEFAUT),
14          _vitesseMax (VITESSE_MAX_DEFAUT),
```

```
15          _estActif(true)
16  {
17      //TODO Auto-generated method stub
18  }
19
20  void ElementActif::init (const XmlConfigFactory::IndexParamValeur&
        indexParamValeur,
21                           XmlConfigFactory& fabrique)
22  {
23      Element::init(indexParamValeur,fabrique);
24  }
25
26  //@uml.annotationsderived_abstraction="platform:/resource/usdp/
        ModeleStructurel.emx#_Pu31cPGuEd-1y9a3HOSRUA?DESTRUCTOR"
27  //@generated "UML to C++ (com.ibm.xtools.transform.uml2.cpp.
        CPPTransformation)"
28  ElementActif::~ElementActif()
29  {
30      //TODO Auto-generated method stub
31  }
32
33  /*void ElementActif::modifierVitesse(double nouvelleVitesse)
34  {
35      _vitesse = nouvelleVitesse;
36  }*/
37
38  void ElementActif::freiner()
39  {
40      if (_estActif)
41      {
42          _estActif = false;
43          emit activationModifiee(false);
44      }
45  }
46
47  void ElementActif::accelerer()
48  {
49      if (!_estActif)
50      {
51          _estActif = true;
52          emit activationModifiee(true);
53      }
54  }
55
56  void ElementActif::definirActivation (bool estActif)
57  {
58      _estActif = estActif;
59  }
60
61  void ElementActif::modifierVitesseMax(double nouvelleVitesseMax)
62  {
63      _vitesseMax = nouvelleVitesseMax;
64  }
65
66
67  bool ElementActif::estActif ()
68  {
69      return _estActif;
70  }
71
72  double ElementActif::vitesse() {
```

```
    return _vitesse;
}

double ElementActif::vitesseMax ()
{
    return _vitesseMax;
}
```

```
#include <QDebug>

#include "modelvols.h"

const char* ModelVols::LABEL_ENTETE[ModelVols::_COUNT] = {
    "Nom",
    "Toboggan"
};

ModelVols::ModelVols(QObject *parent)
:QAbstractTableModel(parent)
{

    _header.push_back(LABEL_ENTETE[kNom]);
}

int ModelVols::rowCount(const QModelIndex&) const
{
    return _data.size();
}

int ModelVols::columnCount(const QModelIndex&) const
{
    return _header.size();
}

QVariant ModelVols::data(const QModelIndex & index, int role) const
{
    if(role == Qt::DisplayRole)
    {
        if(index.column() == kNom)
            return QVariant(_data[index.row()]->nom());
    }
    return QVariant();
}

bool ModelVols::setData(const QModelIndex & index, const QVariant & value,
    int role)
{
//    if(index.column() == kToboggan)
//        _data[index.row()]->associer(value);
}

void ModelVols::ajouterVol(Vol* vol)
{
    beginInsertRows(QModelIndex(), _data.size(), _data.size()+1);
    _data.push_back(vol);
    endInsertRows();
}

void ModelVols::retirerVol(int index)
{
```

```
52      beginRemoveRows(QModelIndex(), index, index);
53      qDebug() << "Retirer Vol :)";
54      _data.remove(index);
55      endRemoveRows();
56  }
57
58  QVariant ModelVols::headerData ( int section, Qt::Orientation orientation,
        int role) const
59  {
60      if(section == kNom)
61          return QVariant(LABEL_ENTETE[kNom]);
62      return QVariant();
63  }
64
65  Vol* ModelVols::at(int index)
66  {
67      if(index < _data.size())
68          return _data[index];
69  }
```

```
1   #include "StrategiePilotageAuto.h"
2   #include "Chariot.h"
3   #include "Toboggan.h"
4   #include "Troncon.h"
5   #include "Tapis.h"
6   #include "Noeud.h"
7
8   StrategiePilotageAuto::StrategiePilotageAuto(Chariot& chariot, Troncon*
        tronconActuel,
9                                                Tapis* tapisAssocie) :
10      StrategiePilotage(chariot,tronconActuel,tapisAssocie)
11  {
12      mettreAJourChemin();
13  }
14
15  StrategiePilotageAuto::StrategiePilotageAuto(const StrategiePilotage&
        modele) :
16          StrategiePilotage(modele)
17  {
18      mettreAJourChemin();
19  }
20
21  void StrategiePilotageAuto::calculerNouveauChemin()
22  {
23      if (_bagage != 0)
24      {
25          _chemin = _bagage->objectifFinal()
26                    ->trouverChemin(_tronconActuel->noeudFin());
27      }
28      else
29      {
30          _chemin = _tapisAssocie->trouverChemin(_tronconActuel->noeudFin());
31      }
32  }
```

```cpp
#include "Toboggan.h"
#include "Bagage.h"
#include "Noeud.h"

Toboggan::Toboggan(const XmlConfigFactory::IndexParamValeur&
    indexParamValeur) :
        Element(indexParamValeur),
        _nombreBagages(0),
        _tronconSupport(0)
{
    //TODO Auto-generated method stub
}

void Toboggan::init (const XmlConfigFactory::IndexParamValeur&
    indexParamValeur,
                    XmlConfigFactory& fabrique)
{
    Element::init(indexParamValeur,fabrique);
    _tronconSupport = dynamic_cast<Troncon*> (fabrique.elementParId(
            indexParamValeur[XmlConfigFactory::NodeName_String[
                XmlConfigFactory::pos]].toInt()));
}

Toboggan::~Toboggan()
{
    //TODO Auto-generated method stub
}

void Toboggan::transfererBagage(Bagage* bagage)
{
    ++_nombreBagages;
    delete bagage;
    emit nombreDeBagages(_nombreBagages);
}


Noeud::Chemin Toboggan::trouverChemin(Noeud* positionActuelle)
{
    return positionActuelle->trouverChemin(_tronconSupport);
}

bool Toboggan::estSupport (const Troncon* troncon) const
{
    return _tronconSupport == troncon;
}

QPointF Toboggan::pointConnexion() const
{
    return _tronconSupport->position();
}

void Toboggan::associerVol(Vol* vol)
{
    _vol = vol;
}

int Toboggan::nombreDeBagages() const
{
    return _nombreBagages;
```

```
}
```

```
#include "Troncon.h"
#include "Noeud.h"

Troncon::Troncon(const XmlConfigFactory::IndexParamValeur& indexParamValeur
    ) :
        Element(indexParamValeur),
        _chariotProprietaire(0),
        _noeudDebut(0),
        _noeudFin(0),
        _estHorsService(false)
{
    //TODO Auto-generated method stub
}


void Troncon::init (const XmlConfigFactory::IndexParamValeur&
    indexParamValeur,
                    XmlConfigFactory& fabrique)
{
    Element::init(indexParamValeur,fabrique);
    _noeudDebut = dynamic_cast<Noeud*> (fabrique.elementParId(
            indexParamValeur[XmlConfigFactory::NodeName_String[
                XmlConfigFactory::noeudDebut]].toInt()));
    _noeudFin = dynamic_cast<Noeud*> (fabrique.elementParId(
            indexParamValeur[XmlConfigFactory::NodeName_String[
                XmlConfigFactory::noeudFin]].toInt()));
    _position = (_noeudDebut->position() + _noeudFin->position()) / 2;
}

Troncon::~Troncon()
{
    //TODO Auto-generated method stub
}

bool Troncon::occuper(Chariot* chariotCandidat)
{
    if (_estHorsService)
    {
        return false;
    }
    else
    {
        if (_chariotProprietaire == 0
            || chariotCandidat == _chariotProprietaire)
        {
            _chariotProprietaire = chariotCandidat;
            return true;
        }
        else
        {
            return false;
        }
    }
}

void Troncon::liberer()
{
    _chariotProprietaire = 0;
```

```
}

bool Troncon::mettreHorsService()
{
    if (_chariotProprietaire == 0)
    {
        _estHorsService = true;
        emit etatModifie();
        return true;
    }
    else
    {
        return false;
    }
}

void Troncon::reparer()
{
    if (_estHorsService)
    {
        _estHorsService = false;
        emit etatModifie();
    }
}

Noeud* Troncon::noeudFin()
{
    return _noeudFin;
}

Noeud* Troncon::noeudDebut()
{
    return _noeudDebut;
}

Troncon::EtatTroncon Troncon::etat()
{
    if (_estHorsService)
    {
        return HORS_SERVICE;
    }
    else if (_chariotProprietaire == 0)
    {
        return LIBRE;
    }
    else
    {
        return OCCUPE;
    }
}
```

```
#include "Vol.h"

Vol::Vol(const QString& nom) : _dateDepart(0), _toboggan(0), _nom(nom)
{
    //TODO Auto-generated method stub
}

Vol::~Vol()
{
```

```
10      //TODO Auto-generated method stub
11  }
12
13  Toboggan* Vol::tobogganAssocie ()
14  {
15      return _toboggan;
16  }
17
18  void Vol::associer(Toboggan* toboggan)
19  {
20      _toboggan = toboggan;
21  }
22
23  QString Vol::nom()
24  {
25      return _nom;
26  }
```