Via Verde

Relatório Final

Mestrado Integrado em Engenharia Informática e Computação

Métodos Formais em Engenharia de Software

Grupo:
João Manuel Azevedo Santos - ei10064
José Carlos da Rocha Lima - ei10012
David Vanhuysse - ei11119

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

14 de Dezembro de 2015

Conteúdo

1 Descrição Informal do Sistema e Lista de Requisitos

O projeto Sistema "Via Verde" foi desenvolvido no âmbito da disciplina de Métodos Formais de Engenharia. Este trabalho tem como objetivo a modelação, na linguagem VDM++, de um sistema de Via Verde. Este é um sistema inovador e criado em Portugal que consiste na realização de pagamentos, fazendo uso deste sistema, em portagens, bombas de gasolina, parques de estacionamento, SCUT's, McDrives, etc.

A via verde é um sistema que se baseia na deteção do carro, por sensores localizados para esse efeito, em determinados locais (portagens, gasolineiras, etc.) e que permite que as pessoas consigam pagar os serviços de maneira mais cómoda e sem necessitarem de sair dos carros. Este sistema baseia-se numa Base de Dados Central, localizada na Via Verde Portugal, que comunica com cada uma das infraestruturas existentes. Cada pessoa que queira possuir um destes aparelhos terá que fornecer os seus dados pessoas bem como o NIB da sua conta pessoal. Depois de validado, o cliente deve colocar o Identificador no vidro da viatura de maneira a que os aparelhos Via Verde o consigam detetar.

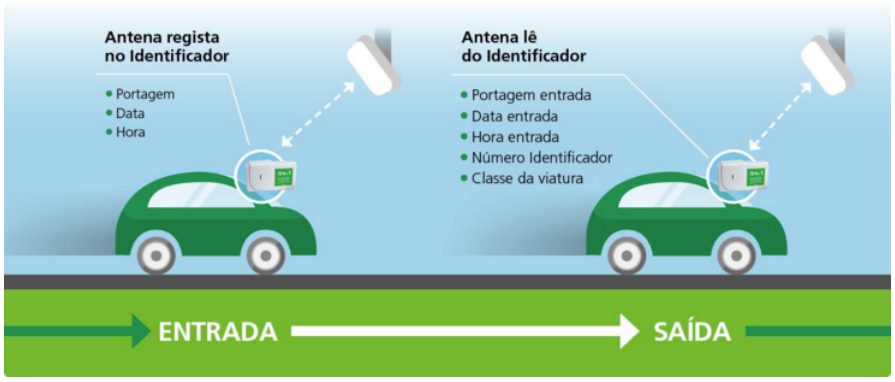

Figura 1: Leitura e registo de dados no identificador.

Depois de analisar todo este processo, o nosso grupo decidiu dividir o projeto nas seguintes classes:

- GreenWayDevice
- GreenWay
- HighwayCosts
- Service_Provider
- User
- Event

Estas classes interagem entre si de modo a criar um Sistema de "Via Verde" virtual o mais aproximado possível do real.

2 Modelo UML

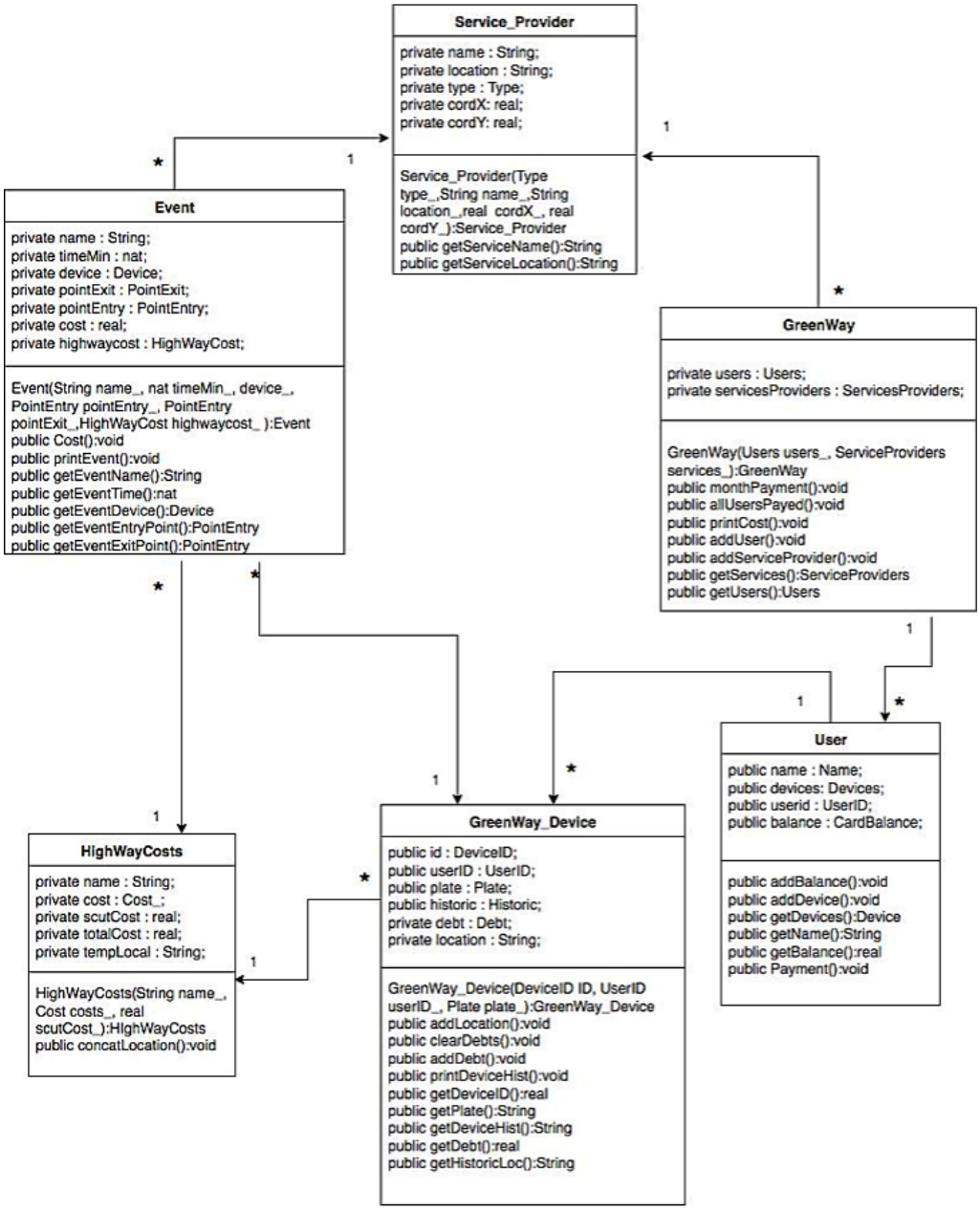

Figura 2: Diagrama UML.

3 Modelo VDM++

3.1 Class GreenWayDevice

Esta classe é responsável por criar e gravar todos os dados contido num identificador e também o guarda o último lugar visitado e determina o montante devido.

```
class GreenWay_Device
types
-- TODO Define types here

public DeviceID = nat1;
public UserID = nat1;
public String = seq of char;
public Plate = String;
public Historic = seq of String;
public Location = String;
public Debt = real;
/*
*/


values
-- TODO Define values here
instance variables
-- TODO Define instance variables here

public id : DeviceID;
public userID : UserID;
public plate : Plate;
public historic : Historic;
private debt : Debt;
private location : String;


operations
-- TODO Define operations here

--construtor
public GreenWay_Device : DeviceID * UserID * Plate ==> GreenWay_Device
  GreenWay_Device(ID, userID_, plate_) == (
  id := ID;
  userID := userID_;
  plate := plate_;
  historic :=[];
  debt := 0;
  return self;
  );

--add location

public addLocation : Event ==> ()
  addLocation(event) == (
  location := event.getEventEntryPoint().getServiceLocation();
  historic := [location]^historic;
  event.Cost(self);
)post historic = [location]^historic~;
```

Figura 3: .

```
public clearDebts: () ==> ()
  clearDebts() == debt :=0
  post debt = 0;

public addDebt : real ==> ()
  addDebt(value) == (
  debt := debt + value;
  )post debt = debt~+value;

--print do historico do device
public printDeviceHist: () ==> ()
printDeviceHist() == (
  IO`println("Historico do Dispositivo:");
  IO`print("Matricula: ");
  IO`println(plate);
  for i in historic
    do(
      IO`println(i);
      );
      IO`println(" ");
);


public getDeviceID: () ==> nat
  getDeviceID() == return id;

public getPlate: () ==> String
  getPlate() == return plate;

public getDeviceHist: () ==> Historic
  getDeviceHist() == return historic;

public getDebt : () ==> real
  getDebt() == return debt;

public getHistoricLoc: () ==> Historic
  getHistoricLoc() == return historic;

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end GreenWay_Device
```

Figura 4: .

3.2 Class GreenWay

Esta classe é responsável por criar e guardar todos os dispositivos e pontos de serviço. Também permite imprimir o montante devido por cada dispositivo.

```
class GreenWay
types
-- TODO Define types here
public User_ = User;
public Users = seq of User_;
public ServiceProvider = Service_Provider;
public ServicesProviders = seq of ServiceProvider;
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
private users : Users;
private servicesProviders : ServicesProviders;
operations
-- TODO Define operations here
public GreenWay: Users * ServicesProviders ==> GreenWay
  GreenWay(users_, services_)== (
    users := users_;
    servicesProviders := services_;
    return self;
  );


-- month payments
public monthPayment: User ==> ()
monthPayment(user_) == user_.allPayed();

public allUsersPayed: () ==> ()
allUsersPayed() == (
  for u in users
  do(
    monthPayment(u);
  )
);
```

Figura 5: .

```
--prints

-- print Costs
public printCost: () ==> ()
 printCost() ==(
  IO`println("Costs from all Devices:");
  for u in users
  do  (
  if len u.getDevices() >= 1 then
  (
  IO`print("User: ");
  IO`println(u.getName());
  for d in u.getDevices()
  do(
    IO`print("Device - ");
    IO`print(d.getDeviceID());
    IO`print(" Plate - ");
    IO`print(d.getPlate());
    IO`print(" to pay: ");
    IO`print(d.getDebt());
    IO`println("€");
    );
    IO`println(" ");
    );
    );
    IO`println(" ");
) ;

public addUser: User_ ==> ()
  addUser(user_) == (
    users := users ^ [user_];
  )
  pre user_ not in set elems users
  post users = users~ ^ [user_];
public addServiceProvider : ServiceProvider ==> ()
addServiceProvider(service_) ==(
servicesProviders := servicesProviders ^[service_]
)pre service_  not in set elems servicesProviders
post servicesProviders = servicesProviders~ ^ [service_];

public getServices: () ==> ServicesProviders
  getServices() == return servicesProviders;
public getUsers: () ==> Users
getUsers() == return users;

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end GreenWay
```

Figura 6: .

3.3 Class HighwayCosts

Esta classe é responsável por criar e guardar os custos de todos os trajetos numa auto-estrada.

```
class HighWayCosts
types
-- TODO Define types here
public String = seq of char;
public Cost_ = map seq of char to real;
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
private name : String;
private cost : Cost_;
private scutCost : real;
private totalCost : real;
private tempLocal : String;

operations
-- TODO Define operations here

public HighWayCosts : String * Cost_ * real ==> HighWayCosts
HighWayCosts(name_, costs_, scutCost_) == (
  name := name_;
  cost := costs_;
  scutCost := scutCost_;
  return self;
)post name = name_ and cost = costs_ and scutCost = scutCost_;

public Costs : String * String ==> real
Costs(local1, local2) ==(
concatLocation(local1, local2);
totalCost := cost(tempLocal);
return totalCost;
);

public concatLocation : String * String ==> ()
concatLocation(local1, local2) == (
tempLocal := local1 ^ local2;
)post tempLocal = local1 ^ local2;

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end HighWayCosts
```

Figura 7: .

3.4 Class Service_Provider

Esta classe identifica um ponto de leitura de identificadores, e é responsável por criar e guardar a informação relevante para estes como a sua localização e o tipo de serviço que disponibilizam.

```
class Service_Provider
types
-- TODO Define types here
public String = seq of char;
public Type = <AutoEstrada> | <Scut> | <Estacionamento>
  | <DriveThru> | <Gasolina>;
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
private name : String;
private location : String;
private type : Type;
private cordX: real;
private cordY: real;


operations
-- TODO Define operations here

public Service_Provider : Type * String * String * real
  * real ==> Service_Provider
Service_Provider(type_,name_,location_, cordX_, cordY_) ==(

name := name_;
type := type_;
location := location_;
cordX := cordX_;
cordY := cordY_;

return self;
)
post name = name_ and type = type_ and location = location_
  and cordX = cordX_ and cordY = cordY_;
```

Figura 8: .

```
--gets

public getServiceName: () ==> String
  getServiceName() == return name;

public getServiceLocation: () ==> String
  getServiceLocation() == return location;

  public getServiceCordX: () ==> real
  getServiceCordX() == return cordX;

  public getServiceCordY: () ==> real
  getServiceCordY() == return cordY;

  public getServiceType: () ==> Type
  getServiceType() == return type;

functions
-- TODO Define functiones here

traces
-- TODO Define Combinatorial Test Traces here
end Service_Provider
```

Figura 9: .

3.5 Class User

Esta classe é responsável por criar e guardar todos os dados de um utilizador, so seus identificadores e o seu saldo. Permite ainda que o utilizador possa carregar o seu saldo bem como efectuar a liquidação da sua dívida.

```
class User
types
-- TODO Define types here

public Name = seq of char;
public UserID = nat1;
public DeviceID = nat1;
public Device = GreenWay_Device;
public Devices = seq of Device;
public CardBalance = real;


values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
public name : Name;
public devices: Devices;
public userid : UserID;
public balance : CardBalance;
operations
-- TODO Define operations here
--construtor
public User : UserID * Name * CardBalance ==> User
  User(ID,name_,cardbalance_) == (

  userid := ID;
  name := name_;
  devices := [];
  balance := cardbalance_;
  return self;
  )
  pre len name_ > 0 and cardbalance_ >= 0
  post userid  = ID and name = name_ and balance = cardbalance_;


  --add money to card
  public addBalance: CardBalance ==> ()
  addBalance(balan) ==(
    balance := balance+balan;
    )
    pre balan > 0
    post balance > 0 and balance > balance~;
```

Figura 10: .

```
  --add money to card
  public addBalance: CardBalance ==> ()
  addBalance(balan) ==(
    balance := balance+balan;
    )
    pre balan > 0
    post balance > 0 and balance > balance~;


  public addDevice: Device ==> ()
  addDevice(device_) ==(

  devices := [device_] ^ devices;

  )
  pre device_ not in set elems devices
  post len devices > len devices~;


  public getDevices : () ==> Devices
    getDevices() == return devices;

    public getName : () ==> Name
    getName() == return name;

    public getBalance : () ==> CardBalance
    getBalance() == return balance;

  public Payment: Device ==> ()
  Payment(device_) ==(
    balance := balance-device_.getDebt();
    device_.clearDebts();
    )
    post balance > 0 and balance < balance~;

    public allPayed:() ==> ()
      allPayed() == (

      for d in devices
      do
      (
        Payment(d);
      )

      );

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end User
```

Figura 11: .

3.6 Class Event

Esta classe é responsável por criar e guardar todos os dados guardados num evento, bem como calcular os pontos de entrada e saída de serviço e os custos das viagens entre pórticos nas auto-estradas.

```
class Event
types
public String = seq of char;
public Device = GreenWay_Device;
public HighWayCost = HighWayCosts;
public Type = <AutoEstrada> | <Estacionamento> | <DriveThru>
  | <Gasolina>;
public PointExit = Service_Provider;
public PointEntry = Service_Provider;

-- TODO Define types here
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
private name : String;
private timeMin : nat;
private device : Device;
private pointExit : PointExit;
private pointEntry : PointEntry;
private cost : real;
private highwaycost : HighWayCost;

operations
-- TODO Define operations here
--caso de pagamento directo
public Event : String * nat * Device * PointEntry * real  ==> Event
Event(name_, timeMin_, device_, pointEntry_, cost_) ==(
name := name_;
timeMin := timeMin_;
device := device_;
pointEntry := pointEntry_;
cost:= cost_;

return self;
)post name = name_ and timeMin = timeMin_ and device = device_ and
  pointEntry = pointEntry_ and cost = cost_;

--caso em que o pagamento depende do ponto de saida
public Event : String * nat * Device * PointEntry * PointExit
  * HighWayCosts ==> Event
Event(name_, timeMin_, device_, pointEntry_, pointExit_,highwaycost_ ) ==(
name := name_;
timeMin := timeMin_;
device := device_;
pointEntry := pointEntry_;
pointExit := pointExit_;
highwaycost := highwaycost_;
cost := 0;
return self;
)post name = name_ and timeMin = timeMin_ and device = device_
  and pointEntry = pointEntry_ and cost = 0 and pointExit = pointExit_;
```

Figura 12: .

```
--calculate Event Cost
public Cost: Device ==> ()
Cost(device_) == (
if pointEntry.getServiceType() = <AutoEstrada> then
  cost := highwaycost.Costs(pointEntry.getServiceLocation(),
    pointExit.getServiceLocation());



  device_.addDebt(cost);

);

public printEvent: () ==> ()
printEvent() ==(
if pointEntry.getServiceType() = <DriveThru> then
  ( IO`println("Event:");
  IO`print("Name: ");
  IO`println(name);
  IO`print("Dispositivo: ");
  IO`println(device.getDeviceID());
  IO`print("Cost: ");
  IO`print(cost);
  IO`println("$");
  IO`print("Point Of Service: ");
  IO`println(pointEntry.getServiceLocation());
  IO`println(" ");
  )
else if pointEntry.getServiceType() = <Estacionamento> then
  ( IO`println("Event:");
    IO`print("Name: ");
    IO`println(name);
    IO`print("Dispositivo: ");
    IO`println(device.getDeviceID());
    IO`print("Cost: ");
    IO`print(cost);
    IO`println("$");
    IO`print("Point Of Service: ");
    IO`println(pointEntry.getServiceLocation());
    IO`println(" ");
  )
else if pointEntry.getServiceType() = <Gasolina> then
  ( IO`println("Event:");
    IO`print("Name: ");
    IO`println(name);
    IO`print("Dispositivo: ");
    IO`println(device.getDeviceID());
    IO`print("Cost: ");
    IO`print(cost);
    IO`println("$");
    IO`print("Point Of Service: ");
    IO`println(pointEntry.getServiceLocation());
```

Figura 13: .

```
    IO`println(" ");
  )
else if pointEntry.getServiceType() = <Scut> then
  ( IO`println("Event:");
    IO`print("Name: ");
    IO`println(name);
    IO`print("Dispositivo: ");
    IO`println(device.getDeviceID());
    IO`print("Cost: ");
    IO`print(cost);
    IO`println("$");
    IO`print("Point Of Service: ");
    IO`println(pointEntry.getServiceLocation());
    IO`println(" ");
  )
else
  (
    IO`println("Event:");
    IO`print("Name: ");
    IO`println(name);
    IO`print("Dispositivo: ");
    IO`println(device.getDeviceID());
    IO`print("Cost: ");
    IO`print(cost);
    IO`println("$");
    IO`print("Entry: ");
    IO`println(pointEntry.getServiceLocation());
    IO`print("Exit: ");
    IO`println(pointExit.getServiceLocation());
    IO`println(" ");
  )
);

public getEventName: () ==> String
  getEventName() == return name;

public getEventTime: () ==> nat
  getEventTime() == return timeMin;

public getEventDevice: () ==> Device
  getEventDevice() == return device;

public getEventEntryPoint: () ==> PointEntry
  getEventEntryPoint() == return pointEntry;

public getEventExitPoint: () ==> PointEntry
  getEventExitPoint() == return pointExit;

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end Event
```

Figura 14: .

4 Validação de Modelos

A cobertura dos testes pode ser observada na secção anterior, representada pelo código destacado a verde.

```
class Testes is subclass of TestCaseUtils
types
-- TODO Define types here
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
operations
-- TODO Define operations here
private teste1: () ==> ()
teste1() == (
--users
dcl u : User := new User(1,"Jose Lima",999999);
dcl u2 : User := new User(1,"John Cena",999999);
dcl u3 : User := new User(1,"Josefino Ferreira Filho",999999);
dcl u4 : User := new User(1,"Nilmar de Jesus",999999);
dcl u5 : User := new User(1,"Washington Silva",999999);
dcl u6 : User := new User(1,"Sofia Ramos",999999);
dcl u7 : User := new User(1,"Andreia Silva",999999);
dcl u8 : User := new User(1,"Ana Ferreira",999999);
dcl u9 : User := new User(1,"Tatiana Silva",999999);
dcl u10 : User := new User(1,"Wellington da Silva Junior",999999);
dcl u11 : User := new User(1,"Rebeca Bueno",999999);
dcl u12 : User := new User(1,"Wesley Souza",999999);
dcl u13 : User := new User(1,"Rodolfo Oliveira Quatro",999999);

--devives
dcl d : GreenWay_Device := new GreenWay_Device(1,1,"25-GG-31");
dcl d1 : GreenWay_Device := new GreenWay_Device(1,1,"39-FH-01");
dcl d2 : GreenWay_Device := new GreenWay_Device(1,1,"17-AA-99");
dcl d3 : GreenWay_Device := new GreenWay_Device(1,1,"38-HL-31");
dcl d4 : GreenWay_Device := new GreenWay_Device(1,1,"68-GP-39");
dcl d5 : GreenWay_Device := new GreenWay_Device(1,1,"69-GP-69");
dcl d6 : GreenWay_Device := new GreenWay_Device(1,1,"21-II-77");
dcl d7 : GreenWay_Device := new GreenWay_Device(1,1,"29-AF-01");
dcl d8 : GreenWay_Device := new GreenWay_Device(1,1,"14-OV-90");
dcl d9 : GreenWay_Device := new GreenWay_Device(1,1,"12-19-VU");
dcl d10 : GreenWay_Device := new GreenWay_Device(1,1,"87-67-RX");
dcl d11 : GreenWay_Device := new GreenWay_Device(1,1,"14-37-BM");
dcl d12 : GreenWay_Device := new GreenWay_Device(1,1,"43-BM-87");
dcl d13 : GreenWay_Device := new GreenWay_Device(1,1,"SV-28-01");

--servicos
dcl sp: Service_Provider :=
new Service_Provider(<AutoEstrada>,"ponto1","LocalA",123,234);
dcl sp2: Service_Provider :=
new Service_Provider(<AutoEstrada>,"ponto2","LocalB",456,567);
dcl sp3: Service_Provider :=
```

```
new Service_Provider(<Estacionamento>,"parque1","LocalB",456,567);
dcl sp4: Service_Provider :=
new Service_Provider(<DriveThru>,"macBoavista","macBoavista",500,489);
dcl sp5: Service_Provider :=
new Service_Provider(<Gasolina>,"galpBoavista","Bomba Galp Boavista",500,489);
dcl sp6: Service_Provider :=
new Service_Provider(<Scut>,"a25 scut Ovar Norte","a25 s. Ovar norte",500,490);

--autoestradas
cl hc : HighWayCosts :=
new HighWayCosts("estrada1",
{"LocalALocalB" |-> 0.6, "LocalBLocalA" |-> 0.6},12);
dcl ev : Event := new Event("AutoEstrada",10,d,sp,sp2,hc);
dcl ev2 : Event := new Event("Estacionamento",60,d,sp3,3);
dcl ev3 : Event := new Event("DriveThru mac", 66, d, sp4,14);
dcl ev4 : Event := new Event("Gasolina", 80, d, sp5, 50);
dcl ev5 : Event := new Event("A25 Scut ovar norte",200, d, sp6,0.5);
dcl greenway: GreenWay := new GreenWay([u],[sp,sp2,sp3,sp4,sp6]);

greenway.addUser(u2);
assertEqual([u,u2],greenway.getUsers());

greenway.addServiceProvider(sp5);

assertEqual([sp,sp2,sp3,sp4,sp6,sp5],greenway.getServices());


u.addDevice(d);
assertEqual([d], u.getDevices());

d.addLocation(ev);
d.addLocation(ev2);
d.addLocation(ev3);




assertEqual([ev3.getEventEntryPoint().getServiceLocation(),
ev2.getEventEntryPoint().getServiceLocation(),
ev.getEventEntryPoint().getServiceLocation()],
d.getHistoricLoc());
d.addLocation(ev4);
d.addLocation(ev5);
assertEqual([ev5.getEventEntryPoint().getServiceLocation(),
ev4.getEventEntryPoint().getServiceLocation(),
ev3.getEventEntryPoint().getServiceLocation(), ev2.getEventEntryPoint().
getServiceLocation(),ev.getEventEntryPoint().getServiceLocation()],
d.getHistoricLoc());

d.printDeviceHist();

ev.printEvent();
```

```
ev2.printEvent();
ev3.printEvent();
ev4.printEvent();
ev5.printEvent();

assertEqual(68.1, d.getDebt());


greenway.printCost();

--pagamento fim do mes
greenway.allUsersPayed();
IO`println("After payment: ");
greenway.printCost();

assertEqual(999999-68.1,u.getBalance());-- saldo apos pagamento

u.addBalance(50);
assertEqual(999999-68.1 + 50,u.getBalance());-- saldo apos carregamento


assertEqual("AutoEstrada",ev.getEventName());
assertEqual(10,ev.getEventTime());
assertEqual(d,ev.getEventDevice());
assertEqual(sp2,ev.getEventExitPoint());
assertEqual([ev5.getEventEntryPoint().getServiceLocation(),
ev4.getEventEntryPoint().getServiceLocation()
,ev3.getEventEntryPoint().getServiceLocation(),
ev2.getEventEntryPoint().getServiceLocation()
,ev.getEventEntryPoint().getServiceLocation()],d.getDeviceHist());
assertEqual("ponto1",sp.getServiceName());
);

public static main: () ==> ()
main() == (
        new Testes().teste1();
);
functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end Testes
```

5 Geração de Código Java

5.1 Class GreenWayDevice

```
package viaverde;

import org.overture.codegen.runtime.*;

import java.util.*;


@SuppressWarnings("all")
public class GreenWay_Device {
    public Number id;
    public Number userID;
    public String plate;
    public VDMSeq historic;
    private Number debt;
    private String location;

    public GreenWay_Device(final Number ID, final Number userID_,
        final String plate_) {
        cg_init_GreenWay_Device_1(ID, userID_, plate_);
    }

    public GreenWay_Device() {
    }

    public void cg_init_GreenWay_Device_1(final Number ID,
        final Number userID_, final String plate_) {
        id = ID;
        userID = userID_;
        plate = plate_;
        historic = SeqUtil.seq();
        debt = 0L;

        return;
    }

    public void addLocation(final Event event) {
        location = event.getEventEntryPoint().getServiceLocation();
        historic = SeqUtil.conc(SeqUtil.seq(location), Utils.copy(historic));

        event.Cost(this);
    }

    public void clearDebts() {
        debt = 0L;
    }

    public void addDebt(final Number value) {
        debt = debt.doubleValue() + value.doubleValue();
```

```
    }

    public void printDeviceHist() {
        IO.println("Historico do Dispositivo:");
        IO.print("Matricula: ");
        IO.println(plate);

        for (Iterator iterator_4 = historic.iterator();
                iterator_4.hasNext();) {
            String i = (String) iterator_4.next();
            IO.println(i);
        }

        IO.println(" ");
    }

    public Number getDeviceID() {
        return id;
    }

    public String getPlate() {
        return plate;
    }

    public VDMSeq getDeviceHist() {
        return Utils.copy(historic);
    }

    public Number getDebt() {
        return debt;
    }

    public VDMSeq getHistoricLoc() {
        return Utils.copy(historic);
    }

    public String toString() {
        return "GreenWay_Device{" + "id := " + Utils.toString(id) +
        ", userID := " + Utils.toString(userID) + ", plate := " +
        Utils.toString(plate) + ", historic := " + Utils.toString(historic) +
        ", debt := " + Utils.toString(debt) + ", location := " +
        Utils.toString(location) + "}";
    }
}
```

5.2 Class GreenWay

```
  package    viaverde;

import org.overture.codegen.runtime.*;

import java.util.*;
```

```
@SuppressWarnings("all")
public class GreenWay {
    private VDMSeq users;
    private VDMSeq servicesProviders;

    public GreenWay(final VDMSeq users_, final VDMSeq services_) {
        cg_init_GreenWay_1(Utils.copy(users_), Utils.copy(services_));
    }

    public GreenWay() {
    }

    public void cg_init_GreenWay_1(final VDMSeq users_, final VDMSeq services_)
        users = Utils.copy(users_);
        servicesProviders = Utils.copy(services_);

        return;
    }

    public void monthPayment(final User user_) {
        user_.allPayed();
    }

    public void allUsersPayed() {
        for (Iterator iterator_1 = users.iterator(); iterator_1.hasNext();) {
            User u = (User) iterator_1.next();
            monthPayment(u);
        }
    }

    public void printCost() {
        IO.println("Costs from all Devices:");

        for (Iterator iterator_2 = users.iterator(); iterator_2.hasNext();) {
            User u = (User) iterator_2.next();

            if (u.getDevices().size() >= 1L) {
                IO.print("User: ");
                IO.println(u.getName());

                for (Iterator iterator_3 = u.getDevices().iterator();
                        iterator_3.hasNext();) {
                    GreenWay_Device d = (GreenWay_Device) iterator_3.next();
                    IO.print("Device - ");
                    IO.print(d.getDeviceID());
                    IO.print(" Plate - ");
                    IO.print(d.getPlate());
                    IO.print(" to pay: ");
                    IO.print(d.getDebt());
                    IO.println("        ");
```

```
                }

                IO.println(" ");
            }
        }

        IO.println(" ");
    }

    public void addUser(final User user_) {
        users = SeqUtil.conc(Utils.copy(users), SeqUtil.seq(user_));
    }

    public void addServiceProvider(final Service_Provider service_) {
        servicesProviders = SeqUtil.conc(Utils.copy(servicesProviders),
                SeqUtil.seq(service_));
    }

    public VDMSeq getServices() {
        return Utils.copy(servicesProviders);
    }

    public VDMSeq getUsers() {
        return Utils.copy(users);
    }

    public String toString() {
        return "GreenWay{" + "users := " + Utils.toString(users) +
        ", servicesProviders := " + Utils.toString(servicesProviders) + "}";
    }
}
```

5.3 Class HighWayCosts

```
package viaverde;

import org.overture.codegen.runtime.*;

import java.util.*;


@SuppressWarnings("all")
public class HighWayCosts {
    private String name;
    private VDMMap cost;
    private Number scutCost;
    private Number totalCost;
    private String tempLocal;

    public HighWayCosts(final String name_, final VDMMap costs_,
        final Number scutCost_) {
        cg_init_HighWayCosts_1(name_, Utils.copy(costs_), scutCost_);
```

```
    }

    public HighWayCosts() {
    }

    public void cg_init_HighWayCosts_1(final String name_, final VDMMap costs_,
        final Number scutCost_) {
        name = name_;
        cost = Utils.copy(costs_);
        scutCost = scutCost_;

        return;
    }

    public Number Costs(final String local1, final String local2) {
        concatLocation(local1, local2);
        totalCost = ((Number) Utils.get(cost, tempLocal));

        return totalCost;
    }

    public void concatLocation(final String local1, final String local2) {
        tempLocal = local1 + local2;
    }

    public String toString() {
        return "HighWayCosts{" + "name := " + Utils.toString(name) +
        ", cost := " + Utils.toString(cost) + ", scutCost := " +
        Utils.toString(scutCost) + ", totalCost := " +
        Utils.toString(totalCost) + ", tempLocal := " +
        Utils.toString(tempLocal) + "}";
    }
}
```

5.4 Class Service_Provider

```
package viaverde;

import org.overture.codegen.runtime.*;

import java.util.*;


@SuppressWarnings("all")
public class Service_Provider {
    private String name;
    private String location;
    private Object type;
    private Number cordX;
    private Number cordY;

    public Service_Provider(final Object type_, final String name_,
```

```
        final String location_, final Number cordX_, final Number cordY_) {
        cg_init_Service_Provider_1(((Object) type_), name_, location_, cordX_,
            cordY_);
    }

    public Service_Provider() {
    }

    public void cg_init_Service_Provider_1(final Object type_,
        final String name_, final String location_, final Number cordX_,
        final Number cordY_) {
        name = name_;
        type = type_;
        location = location_;
        cordX = cordX_;
        cordY = cordY_;

        return;
    }

    public String getServiceName() {
        return name;
    }

    public String getServiceLocation() {
        return location;
    }

    public Number getServiceCordX() {
        return cordX;
    }

    public Number getServiceCordY() {
        return cordY;
    }

    public Object getServiceType() {
        return type;
    }

    public String toString() {
        return "Service_Provider{" + "name := " + Utils.toString(name) +
        ", location := " + Utils.toString(location) + ", type := " +
        Utils.toString(type) + ", cordX := " + Utils.toString(cordX) +
        ", cordY := " + Utils.toString(cordY) + "}";
    }
}
```

5.5 Class User

```
package viaverde;

import org.overture.codegen.runtime.*;

import java.util.*;


@SuppressWarnings("all")
public class User {
    public String name;
    public VDMSeq devices;
    public Number userid;
    public Number balance;

    public User(final Number ID, final String name_, final Number cardbalance_)
        cg_init_User_1(ID, name_, cardbalance_);
    }

    public User() {
    }

    public void cg_init_User_1(final Number ID, final String name_,
        final Number cardbalance_) {
        userid = ID;
        name = name_;
        devices = SeqUtil.seq();
        balance = cardbalance_;

        return;
    }

    public void addBalance(final Number balan) {
        balance = balance.doubleValue() + balan.doubleValue();
    }

    public void addDevice(final GreenWay_Device device_) {
        devices = SeqUtil.conc(SeqUtil.seq(device_), Utils.copy(devices));
    }

    public VDMSeq getDevices() {
        return Utils.copy(devices);
    }

    public String getName() {
        return name;
    }

    public Number getBalance() {
        return balance;
    }
```

```
    public void Payment(final GreenWay_Device device_) {
        balance = balance.doubleValue() - device_.getDebt().doubleValue();
        device_.clearDebts();
    }

    public void allPayed() {
        for (Iterator iterator_5 = devices.iterator(); iterator_5.hasNext();) {
            GreenWay_Device d = (GreenWay_Device) iterator_5.next();
            Payment(d);
        }
    }

    public String toString() {
        return "User{" + "name := " + Utils.toString(name) + ", devices := " +
        Utils.toString(devices) + ", userid := " + Utils.toString(userid) +
        ", balance := " + Utils.toString(balance) + "}";
    }
}
```

5.6 Class Event

```
package viaverde;

import org.overture.codegen.runtime.*;

import java.util.*;


@SuppressWarnings("all")
public class Event {
    private String name;
    private Number timeMin;
    private GreenWay_Device device;
    private Service_Provider pointExit;
    private Service_Provider pointEntry;
    private Number cost;
    private HighWayCosts highwaycost;

    public Event(final String name_, final Number timeMin_,
        final GreenWay_Device device_, final Service_Provider pointEntry_,
        final Number cost_) {
        cg_init_Event_1(name_, timeMin_, device_, pointEntry_, cost_);
    }

    public Event(final String name_, final Number timeMin_,
        final GreenWay_Device device_, final Service_Provider pointEntry_,
        final Service_Provider pointExit_, final HighWayCosts highwaycost_) {
        cg_init_Event_2(name_, timeMin_, device_, pointEntry_, pointExit_,
            highwaycost_);
    }
```

```
public Event() {
}

public void cg_init_Event_1(final String name_, final Number timeMin_,
    final GreenWay_Device device_, final Service_Provider pointEntry_,
    final Number cost_) {
    name = name_;
    timeMin = timeMin_;
    device = device_;
    pointEntry = pointEntry_;
    cost = cost_;

    return;
}

public void cg_init_Event_2(final String name_, final Number timeMin_,
    final GreenWay_Device device_, final Service_Provider pointEntry_,
    final Service_Provider pointExit_, final HighWayCosts highwaycost_) {
    name = name_;
    timeMin = timeMin_;
    device = device_;
    pointEntry = pointEntry_;
    pointExit = pointExit_;
    highwaycost = highwaycost_;
    cost = 0L;

    return;
}

public void Cost(final GreenWay_Device device_) {
    if (Utils.equals(pointEntry.getServiceType(),
                viaverde.quotes.AutoEstradaQuote.getInstance())) {
        cost = highwaycost.Costs(pointEntry.getServiceLocation(),
                pointExit.getServiceLocation());
    }

    device_.addDebt(cost);
}

public void printEvent() {
    if (Utils.equals(pointEntry.getServiceType(),
                viaverde.quotes.DriveThruQuote.getInstance())) {
        IO.println("Event:");
        IO.print("Name: ");
        IO.println(name);
        IO.print("Dispositivo: ");
        IO.println(device.getDeviceID());
        IO.print("Cost: ");
        IO.print(cost);
        IO.println("$");
        IO.print("Point Of Service: ");
        IO.println(pointEntry.getServiceLocation());
```

```
            IO.println(" ");
        } else {
            if (Utils.equals(pointEntry.getServiceType(),
                    viaverde.quotes.EstacionamentoQuote.getInstance())) {
                IO.println("Event:");
                IO.print("Name: ");
                IO.println(name);
                IO.print("Dispositivo: ");
                IO.println(device.getDeviceID());
                IO.print("Cost: ");
                IO.print(cost);
                IO.println("$");
                IO.print("Point Of Service: ");
                IO.println(pointEntry.getServiceLocation());
                IO.println(" ");
            } else {
                if (Utils.equals(pointEntry.getServiceType(),
                        viaverde.quotes.GasolinaQuote.getInstance())) {
                    IO.println("Event:");
                    IO.print("Name: ");
                    IO.println(name);
                    IO.print("Dispositivo: ");
                    IO.println(device.getDeviceID());
                    IO.print("Cost: ");
                    IO.print(cost);
                    IO.println("$");
                    IO.print("Point Of Service: ");
                    IO.println(pointEntry.getServiceLocation());
                    IO.println(" ");
                } else {
                    if (Utils.equals(pointEntry.getServiceType(),
                            viaverde.quotes.ScutQuote.getInstance())) {
                        IO.println("Event:");
                        IO.print("Name: ");
                        IO.println(name);
                        IO.print("Dispositivo: ");
                        IO.println(device.getDeviceID());
                        IO.print("Cost: ");
                        IO.print(cost);
                        IO.println("$");
                        IO.print("Point Of Service: ");
                        IO.println(pointEntry.getServiceLocation());
                        IO.println(" ");
                    } else {
                        IO.println("Event:");
                        IO.print("Name: ");
                        IO.println(name);
                        IO.print("Dispositivo: ");
                        IO.println(device.getDeviceID());
                        IO.print("Cost: ");
                        IO.print(cost);
                        IO.println("$");
```

```
                        IO.print("Entry: ");
                        IO.println(pointEntry.getServiceLocation());
                        IO.print("Exit: ");
                        IO.println(pointExit.getServiceLocation());
                        IO.println(" ");
                    }
                }
            }
        }
    }

    public String getEventName() {
        return name;
    }

    public Number getEventTime() {
        return timeMin;
    }

    public GreenWay_Device getEventDevice() {
        return device;
    }

    public Service_Provider getEventEntryPoint() {
        return pointEntry;
    }

    public Service_Provider getEventExitPoint() {
        return pointExit;
    }

    public String toString() {
        return "Event{" + "name := " + Utils.toString(name) + ", timeMin := " +
        Utils.toString(timeMin) + ", device := " + Utils.toString(device) +
        ", pointExit := " + Utils.toString(pointExit) + ", pointEntry := " +
        Utils.toString(pointEntry) + ", cost := " + Utils.toString(cost) +
        ", highwaycost := " + Utils.toString(highwaycost) + "}";
    }
}
```

6 Conclusão

Não obstante algumas dificuldades que se nos foram apresentando, nomeadamente na construção da estrutura de dados, a qual foi necessário repensar e reestruturar, o projeto que que nos foi proposto foi concluído com sucesso. Uma vez compreendidas e ultrapassadas estas dificuldades, a implementação decorreu sem grandes percalços.

Inicialmente pretendiamos implementar a funcionalidade de verificar se , arbitrando uma velocidade máxima, a viagem entre dois pontos seria possível. Isto permitiria detectar por exemplo, dispositivos copiados. Esta acabou por nao ser implementada.

O esforço dedicado a este projecto foi dividido de forma equitativa pelos seus elementos.