Mestrado Integrado em Engenharia Informática e Computação

Métodos Formais em Engenharia de Software
4º ano 1º semestre

”Green Way”System

Relatório Final

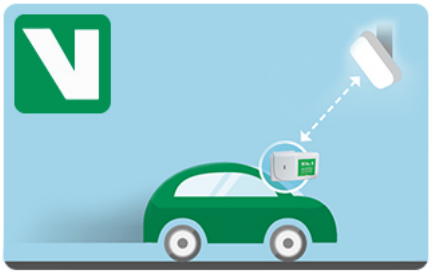
Grupo 4MIEIC03_T13:
Maria João Marques - 201204979 - ei12104@fe.up.pt
Sofia Oliveira Reis - 201200742 - ei12041@fe.up.pt

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

15 de Dezembro de 2015

1 Descrição Informal do Sistema e Lista de Requisitos

1.1 Descrição Informal do Sistema



O *"Green Way"System*, ou em português Via Verde, presta serviços de gestão de sistemas eletrónicos de cobrança por utilização de infraestruturas rodoviárias e de outras utilizadas por veículos automóveis, tais como auto-estradas, parques de estacionamento, bombas de gasolina, *ferries*, *McDrives*, etc.

Para aderir ao sistema os utilizadores necessitam de obter um equipamento conectado ao seu cartão de crédito/débito que deve estar fixo ao pára-brisas do respetivo veículo.

Ao fim do mês são acumulados todos os gastos do utilizador e descontados da sua conta, se não este não tiver saldo suficiente a partir desse momento passa a não conseguir usar o serviço.

1.2 Lista de Requisitos

Id	Prioridade	Descrição
R1	Obrigatório	Fornecedores de serviços devem estabelecer um protocolo com o *manager* da Via Verde
R2	Obrigatório	Os custos são mensalmente debitados por transferência bancária
R3	Obrigatório	Deve incluir uma forma de calcular tarifas das auto-estradas baseado em tabelas configuráveis
R4	Obrigatório	Deve estar preparado para detetar e alertar anomalias e fraude
R5	Obrigatório	Autoridades de segurança ou legais devem ter acesso à localização do veículo como também ao trajeto efetuado

2 Modelo Visual de UML

2.1 Modelo de Casos de Utilização

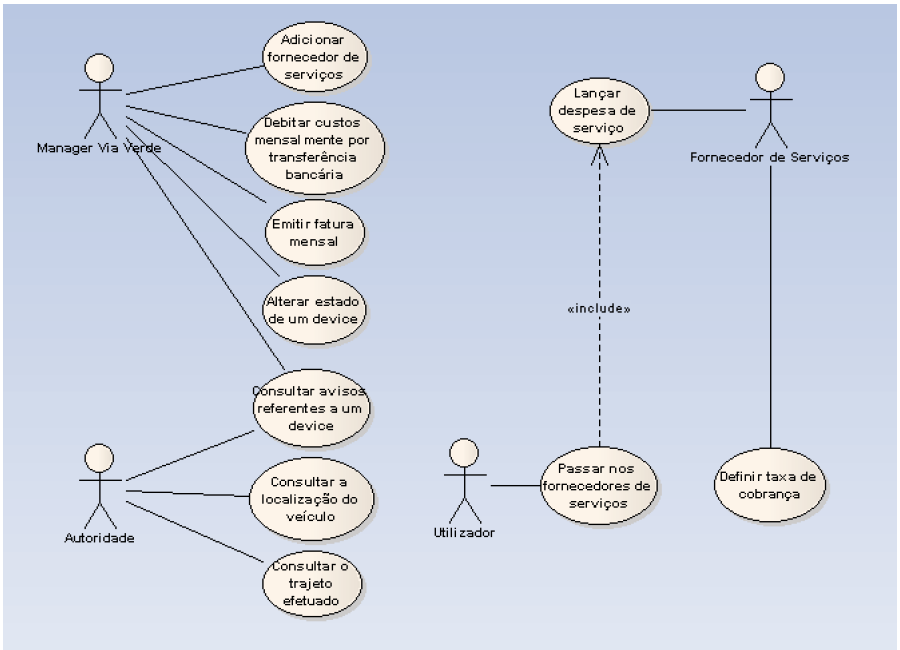


Figura 1: Modelo de casos de utilização

Os cenários dos casos de utilização principais são descritos de seguida:

Cenário	**Adicionar fornecedor de serviços**
Descrição	Cenário normal em que o *manager* da Via Verde adiciona um fornecedor de serviços (seleciona o tipo e a localização)
Pré-condições	1. O novo fornecedor ainda não existir na *Greenway*
Pós-condições	1. O número de elementos calculado no contador de serviços tem de ser igual ao guardado no *set* de serviços registados na Via Verde

Cenário	**Debitar custos mensalmente por transferência bancária**
Descrição	No final do mês são debitados todos os custos acumulados pelo utilizador do sistema ao longo do mês)
Pré-condições	1. O device tem de estar registado no sistema
Pós-condições	1. O número de elementos calculado no contador de serviços tem de ser igual ao guardado no *set* de serviços registados na Via Verde
Exceções	1. Se não tiver dinheiro suficiente fica impossibilitado de usar o serviço

Cenário	**Passar nos fornecedores de serviços**
Descrição	Cenário normal em que o utilizador usa o serviço num dos fornecedores de serviços
Pré-condições	1. O device tem de estar registado no sistema
Pós-condições	1. O número de elementos calculado no contador de serviços tem de ser igual ao guardado no *set* de serviços registados na Via Verde
Exceções	1. O utilizador não possui dinheiro suficiente na sua conta

Cenário	**Consultar avisos referentes a um *device***
Descrição	Tanto como manager da Via Verde e como autoridade quero consultar avisos referentes ao device de um utilizador)
Pré-condições	1. O serviço tem de fazer parte dos serviços registados
Exceções	1. O *device* está *disable* ou não faz parte dos *devices* registados 2. O fornecedor de serviços não está no mesmo sítio que o ponto de leitura 3. A matrícula lida é diferente da matrícula associada ao *device*

2.2 Modelo de Classes

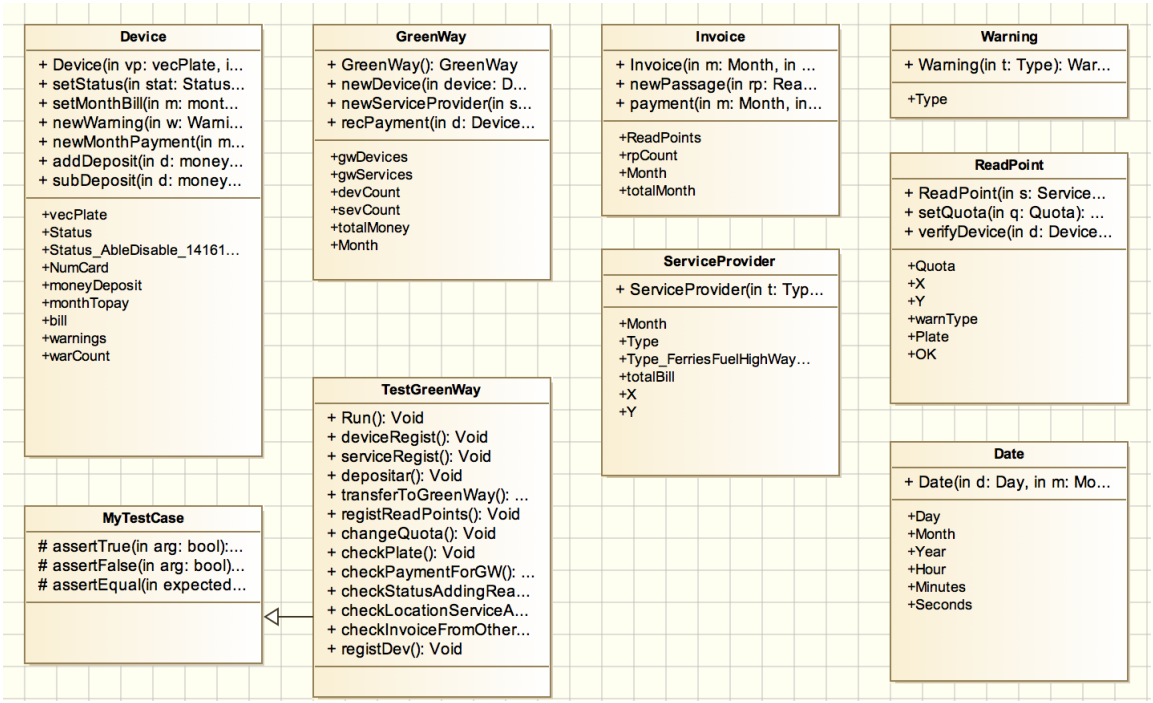


Figura 2: UML de classes

Classe	Descrição
Device	Define o aparelho usado para identificar o veículo durante a passagem em serviços.
Date	Define a data e a hora do registo da passagem por um serviço.
GreenWay	Define a via verde em geral. Possui registo de aparelhos e serviços.
Invoice	Define uma fatura que possui o registo de todas as passagens de um certo device.
ServiceProvider	Define um Serviço, o tipo de serviço e a sua localização, por exemplo, McDrive, Auto-Estrada, etc.
ReadPoint	Define ponto de passagem que possui uma localização, serviço associado, aparelho que passa, preço e data de passagem.
Warning	Define um aviso e o seu tipo.
TestGreenWay	Define os testes para o Sistema.

3 Modelo formal VDM++

Para visualizar o coverage do código com cores é necessário aceder à pasta *coverage* na pasta do projeto.

3.1 *Class GreenWay*

```
class GreenWay
types
  public gwDevices = set of Device;
  public gwServices = set of ServiceProvider;
  public devCount = nat;
  public sevCount = nat;
  public totalMoney = real;
  public Month = int;

values
-- TODO Define values here
instance variables
 static public gwdevices : gwDevices := {};
 static public gwservices : gwServices := {};

 public devicesCount : devCount := card gwdevices;
 public servicesCount : sevCount := card gwservices;
 public total : totalMoney;
operations

 -- GreenWay Constructor
 public GreenWay : () ==> GreenWay
 GreenWay() == (

 total := 0;
  return self);
```

```
 -- Regist device on green way
 public newDevice: Device ==> ()
 newDevice(device) == (


  gwdevices := {device} union gwdevices;
  devicesCount := devicesCount + 1;
  device.setStatus(<Able>);
  IO`println("New device registed!");)
 pre(device not in set gwdevices)
 post(devicesCount = card gwdevices);

 -- Regist service on green way - R1: Estabelecer Protocolo Com a GreenWay
 public newServiceProvider: ServiceProvider ==> ()
 newServiceProvider(service) == (
  gwservices := {service} union gwservices;
  servicesCount := servicesCount + 1;
  IO`println("New service registed!");
 )

 pre(service not in set gwservices)
 post(servicesCount = card gwservices);

 -- Recive Payment of a device in a month - R2
 public recPayment: Device * Month ==> ()
 recPayment(d,m) == (
   if(d.moneyDep - d.monthBill(m) < 0) then d.setStatus(<Disable>)
   else(
     total := total + d.monthBill(m);
     d.setMonthBill(m,0);
     d.setStatus(<Able>);))

    pre(d in set gwdevices);


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

3.2 *Class ServiceProvider*

```
class ServiceProvider
types
 public Month = seq1 of char;
 public Type = <HighWay> | <ParkingLot> | <Fuel> | <Ferries> | <McDrive> | <None>;
 public totalBill  = rat;
 public X = real;
 public Y = real;

instance variables
 public type : Type := <None>;
 public xs : X;



 public ys : Y;
operations
  -- Constructor Service Provider
```

```
  public ServiceProvider: Type * X * Y ==> ServiceProvider
  ServiceProvider(t, x, y) == (
   type := t;
   xs := x;
   ys := y;
  );

end ServiceProvider
```

3.3 *Class Device*

```
class Device
types
 public vecPlate = seq1 of char;
 public Status = <Able> | <Disable>;
 public NumCard = int;
 public moneyDeposit = real;
 public monthTopay = int;
 public bill = rat;
 public warnings = set of Warning;
 public warCount = nat;


instance variables
 static public warns : warnings := {};
 public wcount : warCount := card warns;
 public plate: vecPlate;
 public cardN: NumCard;

 public status: Status := <Disable>;
 public moneyDep : moneyDeposit;
 public monthBill: map monthTopay to bill := {1|->0,2|->0,
                     3|->0, 4|->0,5|->0,
                     6|->0,7|->0,8|->0,
                     9|->0, 10|->0,11|->0,
                     12|->0};


operations


 -- Device Constructor
 public Device: vecPlate * NumCard * moneyDeposit  ==> Device
 Device(vp, nc, md) == (
   plate := vp;
   cardN := nc;
   moneyDep := md;
   return self

 );


 -- Set Device Status, when registed on green way, for example
 public setStatus: Status ==> ()

 setStatus(stat) ==
   status := stat;

   -- Set Month Bill
   public setMonthBill: monthTopay * bill ==> ()
    setMonthBill(m, q) ==
```

```
    monthBill(m) := q;

   -- Create new warning
    public newWarning: Warning ==> ()


 newWarning(w) == (
  warns := {w} union warns;
  wcount := wcount + 1;
  IO`println("New warning registed!");
 )
 post(wcount = card warns);

 -- add payment of a certain month
 public newMonthPayment: monthTopay * bill ==> ()
 newMonthPayment(m,b) ==
 (
  monthBill(m) := monthBill(m) + b;
 );

  -- add new deposit

 public addDeposit: moneyDeposit ==> ()
 addDeposit(d) ==
 (
  moneyDep := moneyDep + d;
 );


 public subDeposit: moneyDeposit ==> ()
 subDeposit(d) ==
 (
  moneyDep := moneyDep - d;
 );

end Device
```

3.4 *Class ReadPoint*

```
class ReadPoint
types
 public Quota = real;
 public X = real;
 public Y = real;
 public warnType = seq1 of char;
 public Plate = seq1 of char;
 public OK = nat;

instance variables

 public quota : Quota;



 public service : ServiceProvider;
 public device : Device;
 public date : Date;
 public xr : X;
 public yr : Y;
 public w1 : Warning;
 public plate : Plate;
```

```
 public ok : OK;
operations

 -- Constructor ReadPoint
 public ReadPoint: ServiceProvider * Quota * Device * Date * X * Y * Plate ==> ReadPoint
 ReadPoint(s,q, d, dat,x,y, p) ==
 (
  service := s;
  quota := q;
  device := d;
  date := dat;

  xr := x;
  yr := y;
  plate := p;
  verifyDevice(d);
  return self;
 );



 -- Changue quota to charge - R3
  public setQuota: Quota ==> ()
   setQuota(q) ==
   quota := q;

 -- Detect Anomalies and fraud - R4
 public verifyDevice: Device ==> ()
 verifyDevice(d) == (
    if d.status = <Disable> or (d not in set GreenWay`gwdevices)
    then(
       IO`println("Device disabled using services");
       w1 := new Warning("Device disabled using services");
       d.newWarning(w1);
       ok := 0;
    )
    else if service.xs <> xr and service.ys <> yr
    then(
      IO`println("The service Provider is not in the same place as the read point");
       w1 := new Warning("The service Provider is not in the same place as the read point");
       d.newWarning(w1);
       ok := 0;
    )
    else if d.plate <> plate
    then(
      IO`println("Device plate different than read plate");
       w1 := new Warning("Device plate different than read plate");
       d.newWarning(w1);
       ok := 0;
    )
    else(
     IO`println("Tudo OK!");
     ok := 1;
    )
 )
 pre(device in set GreenWay`gwdevices and service in set GreenWay`gwservices);

end ReadPoint
```

3.5 *Class Date*

```
class Date
types
  public Day = nat1;
  public Month = nat1;
  public Year = int;
  public Hour = int;
  public Minutes = int;
  public Seconds = int;

instance variables
  public day : Day;
  public month : Month;
  public year : Year;
  public hour : Hour;
  public minutes : Minutes;
  public seconds : Seconds;



  inv day <= 31 and
    month <= 12 and
     if month in set {4, 6, 9, 11}
     then day <= 30
     else (month = 2) => (day <= 29);

   inv hour < 24 and
    minutes < 60 and
    seconds < 60;

operations
   -- Date Constructor
   public Date : Day * Month * Year * Hour * Minutes * Seconds ==> Date
   Date(d,m,y,h,min,sec) == (
   day := d;
   month := m;
   year := y;
   hour := h;
   minutes := min;
   seconds := sec;
  return self
  );
end Date
```

3.6 *Class Warning*

```
class Warning
types
  public Type = seq1 of char;
instance variables
  public type : Type;
operations
  -- Constructor Warning
  public Warning: Type  ==> Warning


 Warning(t) == (
   type := t;
   return self
 );
end Warning
```

4 Validação do Modelo

4.1 Class MyTestCase

```
class MyTestCase
/*
  Superclass for test classes, simpler but more practical than VDMUnit`TestCase.
  For proper use, you have to do: New -> Add VDM Library -> IO.
  JPF, FEUP, MFES, 2014/15.
*/

operations

 -- Simulates assertion checking by reducing it to pre-condition checking.
 -- If 'arg' does not hold, a pre-condition violation will be signaled.

 protected assertTrue: bool ==> ()
 assertTrue(arg) ==
  return
 pre arg;


 protected assertFalse: bool ==> ()
 assertFalse(arg) ==
  return
 pre not arg;

 -- Simulates assertion checking by reducing it to post-condition checking.
 -- If values are not equal, prints a message in the console and generates
 -- a post-conditions violation.

 protected assertEqual: ? * ? ==> ()
 assertEqual(expected, actual) ==
  if expected <> actual then (
    IO`print("Actual value (");
    IO`print(actual);
    IO`print(") different from expected (");
    IO`print(expected);
    IO`println(")\n")
  )
 post expected = actual

end MyTestCase
```

4.2 Class TestGreenWay

```
class TestGreenWay is subclass of MyTestCase

operations

  public Run : () ==> ()
   Run() == (
    -- Registar Device
    deviceRegist();
    -- Registar Service
    serviceRegist();
    -- Depositar
    depositar();
    -- Transfer
```

```
  transferToGreenWay();
  -- create Invoice
  registReadPoints();
  -- Change Quota
  changeQuota();
  -- Check Plate
  checkPlate();
  -- Check Payment For GW
  checkPaymentForGW();
  -- Check Status Disable when add Read Point
  checkStatusAddingReadPoint();
  -- Check diferent locations of Service e Read Point at the same time
  checkLocationServiceAndRP();
  -- Check invoide from other month

  checkInvoiceFromOtherMonth();
  -- Check error recpeyment device when is not registed in plataform
  registDev();



 );

public deviceRegist : () ==> ()
deviceRegist() == (
dcl gw : GreenWay;
dcl d1 : Device;
dcl d2 : Device;
IO`println("---Test : Create Device ---");

gw := new GreenWay();

d1 := new Device("00-BF-34",107893789,1500);
d2 := new Device("01-FF-34",107835628,2000);

gw.newDevice(d1);
assertTrue(gw.devicesCount = 1);
gw.newDevice(d2);

assertTrue(gw.devicesCount = 2);
);

public serviceRegist : () ==> ()
serviceRegist() == (
dcl gw : GreenWay;
dcl s1 : ServiceProvider;
dcl s2 : ServiceProvider;
IO`println("---Test : Create Service ---");

gw := new GreenWay();
s1 := new ServiceProvider(<HighWay>,1.89,2.83);

s2 := new ServiceProvider(<ParkingLot>,10.59,4.65);

gw.newServiceProvider(s1);
assertTrue(gw.servicesCount = 1);
gw.newServiceProvider(s2);
assertTrue(gw.servicesCount = 2);
);


public depositar : () ==> ()
depositar() == (
dcl d1 : Device;
```

```
IO`println("---Test : Deposite ---");

d1 := new Device("00-BF-34",107893789,1500);
d1.addDeposit(500);
assertTrue(d1.moneyDep = 2000);
);

public transferToGreenWay : () ==> ()
transferToGreenWay() == (

dcl gw : GreenWay;
dcl d1 : Device;
IO`println("---Test : Transfer ---");

d1 := new Device("00-BF-34",107893789,1500);
gw := new GreenWay();
gw.newDevice(d1);
d1.newMonthPayment(2, 750);
assertTrue(d1.monthBill(2) = 750);
gw.recPayment(d1,2);
d1.subDeposit(750);
assertTrue(gw.total = 750);
assertTrue(d1.monthBill(2) = 0);
assertTrue(d1.moneyDep = 750);
);


public registReadPoints : () ==> ()
registReadPoints() == (
dcl gw : GreenWay;
dcl s1 : ServiceProvider;
dcl s2 : ServiceProvider;
dcl d1 : Device;
dcl d2 : Device;
dcl dat1 : Date;
dcl dat2 : Date;
dcl rd1 : ReadPoint;
dcl rd2 : ReadPoint;
dcl iv1 : Invoice;
IO`println("---Test : Regist Read Points ---");
gw := new GreenWay();
d1 := new Device("00-BF-34",107893789,1500);
d2 := new Device("10-FF-34",107723682,2000);
gw.newDevice(d1);
gw.newDevice(d2);

s1 := new ServiceProvider(<HighWay>,1.89,2.83);
s2 := new ServiceProvider(<ParkingLot>,10.59,4.65);
gw.newServiceProvider(s1);
gw.newServiceProvider(s2);
dat1 := new Date(14,12,2015, 21,35,39);
dat2 := new Date(14,12,2015, 21,40,40);
rd1 := new ReadPoint(s1,10,d1, dat1,1.89, 2.83, "00-BF-34");
rd2 := new ReadPoint(s2,15,d1, dat2, 10.59,4.65, "00-BF-34");
iv1 := new Invoice(12,d1);
iv1.newPassage(rd1);
iv1.newPassage(rd2);

assertTrue(iv1.passagesCount = 2);
assertTrue(iv1.total = 25);
);


 public changeQuota : () ==> ()
changeQuota() == (
```

```
    dcl gw : GreenWay;
    dcl s1 : ServiceProvider;
    dcl d1 : Device;
    dcl dat1 : Date;
    dcl rd1 : ReadPoint;
    IO`println("---Test : Change Quota ---");

    gw := new GreenWay();
   d1 := new Device("00-BF-34",107893789,1500);
   gw.newDevice(d1);
   s1 := new ServiceProvider(<HighWay>,1.89,2.83);
   gw.newServiceProvider(s1);
   dat1 := new Date(14,12,2015, 21,35,39);
   rd1 := new ReadPoint(s1,10,d1, dat1,1.89, 2.83, "00-BF-34");

   rd1.setQuota(20);
   assertTrue(rd1.quota = 20);
   );

   -- EXCEPTIONS TESTS
   public checkPlate : () ==> ()
   checkPlate() == (
    dcl gw : GreenWay;
    dcl s1 : ServiceProvider;
    dcl d1 : Device;
    dcl dat1 : Date;
    dcl rd1 : ReadPoint;
    IO`println("---Test : Check Plate ---");
    gw := new GreenWay();
   d1 := new Device("00-FF-34",107893789,1500);

   gw.newDevice(d1);
   s1 := new ServiceProvider(<HighWay>,1.89,2.83);
   gw.newServiceProvider(s1);
   dat1 := new Date(14,12,2015, 21,35,39);
   rd1 := new ReadPoint(s1,10,d1, dat1,1.89, 2.83, "00-BF-34");

   );

  public checkPaymentForGW : () ==> ()
  checkPaymentForGW() == (
    dcl gw : GreenWay;
  dcl s1 : ServiceProvider;
  dcl s2 : ServiceProvider;
  dcl d1 : Device;
  dcl dat1 : Date;
  dcl dat2 : Date;
  dcl rd1 : ReadPoint;
  dcl rd2 : ReadPoint;
  dcl iv1 : Invoice;
  IO`println("---Test : Possible or not possible Transfer for gw ---");
  gw := new GreenWay();
  d1 := new Device("00-BF-34",107893789,10);
  gw.newDevice(d1);
  s1 := new ServiceProvider(<HighWay>,1.89,2.83);
  s2 := new ServiceProvider(<ParkingLot>,10.59,4.65);
  gw.newServiceProvider(s1);
  gw.newServiceProvider(s2);
  dat1 := new Date(14,12,2015, 21,35,39);
  dat2 := new Date(14,12,2015, 21,40,40);
  rd1 := new ReadPoint(s1,10,d1, dat1,1.89, 2.83, "00-BF-34");
  rd2 := new ReadPoint(s2,15,d1, dat2, 10.59,4.65, "00-BF-34");
  iv1 := new Invoice(12,d1);
  iv1.newPassage(rd1);
  iv1.newPassage(rd2);
```

```
 iv1.payment(12,iv1.total);
 assertTrue(d1.moneyDep = 10);
 assertTrue(d1.monthBill(12) = 25);


 gw.recPayment(d1, 12);
 assertTrue(d1.status = <Disable>);

);


public checkStatusAddingReadPoint : () ==> ()
  checkStatusAddingReadPoint() == (
   dcl gw : GreenWay;
   dcl s1 : ServiceProvider;
   dcl d1 : Device;
   dcl dat1 : Date;
   dcl rd1 : ReadPoint;
   IO`println("---Test : Check Status Disable when add Read Point ---");
   gw := new GreenWay();
  d1 := new Device("00-FF-34",107893789,1500);
  gw.newDevice(d1);
  d1.setStatus(<Disable>);

  s1 := new ServiceProvider(<HighWay>,1.89,2.83);
  gw.newServiceProvider(s1);
  dat1 := new Date(14,12,2015, 21,35,39);
  rd1 := new ReadPoint(s1,10,d1, dat1,1.89, 2.83, "00-BF-34");
  );

 public checkLocationServiceAndRP : () ==> ()
  checkLocationServiceAndRP() == (
   dcl gw : GreenWay;
   dcl s1 : ServiceProvider;
   dcl d1 : Device;
   dcl dat1 : Date;
   dcl rd1 : ReadPoint;
   IO`println("---Test : Check Location Service and RP ---");
   gw := new GreenWay();
  d1 := new Device("00-FF-34",107893789,1500);
  gw.newDevice(d1);
  s1 := new ServiceProvider(<HighWay>,1.89,2.83);
  gw.newServiceProvider(s1);
  dat1 := new Date(14,12,2015, 21,35,39);
  rd1 := new ReadPoint(s1,10,d1, dat1,1.80, 2.84, "00-FF-34");
  );



   -- ERROR TESTS
  public checkInvoiceFromOtherMonth : () ==> ()
  checkInvoiceFromOtherMonth() == (
   dcl gw : GreenWay;
   dcl s1 : ServiceProvider;
   dcl d1 : Device;
   dcl dat1 : Date;
   dcl rd1 : ReadPoint;
   dcl iv1 : Invoice;
   IO`println("---Test : Check Invoice From Other Month ---");
   gw := new GreenWay();
  d1 := new Device("00-FF-34",107893789,1500);
  gw.newDevice(d1);
  s1 := new ServiceProvider(<HighWay>,1.89,2.83);
  gw.newServiceProvider(s1);
```

```
   dat1 := new Date(14,11,2015, 21,35,39);
   rd1 := new ReadPoint(s1,10,d1, dat1,1.89, 2.83, "00-FF-34");
   iv1 := new Invoice(12,d1);

  -- assertTrue(rd1.date.month = iv1.month);

   );



 public registDev : () ==> ()
  registDev() == (
  dcl gw : GreenWay;
  dcl d1 : Device;
  dcl d2 : Device;
  IO`println("---Test : Regist Device ---");

  gw := new GreenWay();
  d1 := new Device("00-BF-34",107893789,1500);
  d2 := new Device("01-FF-34",107835628,2000);
  --gw.recPayment(d1,12);
  );

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

5 Verificação do Modelo

5.1 Exemplo de verificação com invariante

Usamos *inv* para verificar dias dependendo do mês, meses (1-12), horas(0-24), minutos(0-60) e segundos(0-60).

```
inv day <= 31 and
                month <= 12 and
                        if month in set {4, 6, 9, 11}
                        then day <= 30
                        else (month = 2) => (day <= 29);

          inv hour < 24 and
                  minutes < 60 and
                  seconds < 60;
```

6 Geração de código

Foi possível gerar o código Java corretamente, mas já não tivemos tempo de gerar testes para testar. Caso tivéssemos tempo, iríamos criar uma interface de texto em Java para ser mais fácil de utilizar a plataforma e testas as funcionalidades. De qualquer forma, com os testes feito em VDM++ foi-nos possível testar todas as funcionalidades.

7 Conclusões

O modelo realizado cobre todos os requisitos mencionado no ponto 1. É um trabalho bastante interessante, visto que, a via verde é cada vez mais utilizado em mais serviços.

Poderíamos acrescentar mais pre e pos condições, criar uma interface em Java para melhor testar o programa. Fazer diferentes tipos de pagamentos para cada serviço.

Os elementos do grupo trabalharam de forma equivalente, sendo que, cada um participou na criação de código e relatório.

8 References

1. Overture tool web site, http://overturetool.org
2. VDM-10 Language Manual, http://kurser.iha.dk/eit/tivdm2/VDM10_lang_man.pdf