Modelado y Diseño del Software

Memoria de la Práctica 1

Participantes:

Nombre	Apellidos	Correo
Pau/Pablo	Fernández Rueda	pauf@uma.es
Eduardo	García Rivas	eduardogarr@uma.es
Francisco Eloy	González Castillo	eloygonzalez@uma.es
María Paulina	Ordóñez Walkowiak	mpow@uma.es
Javier	Toledo Delgado	javier.toledo.delgado@uma.es
Daniela	Suárez Morales	danielasuarez@uma.es

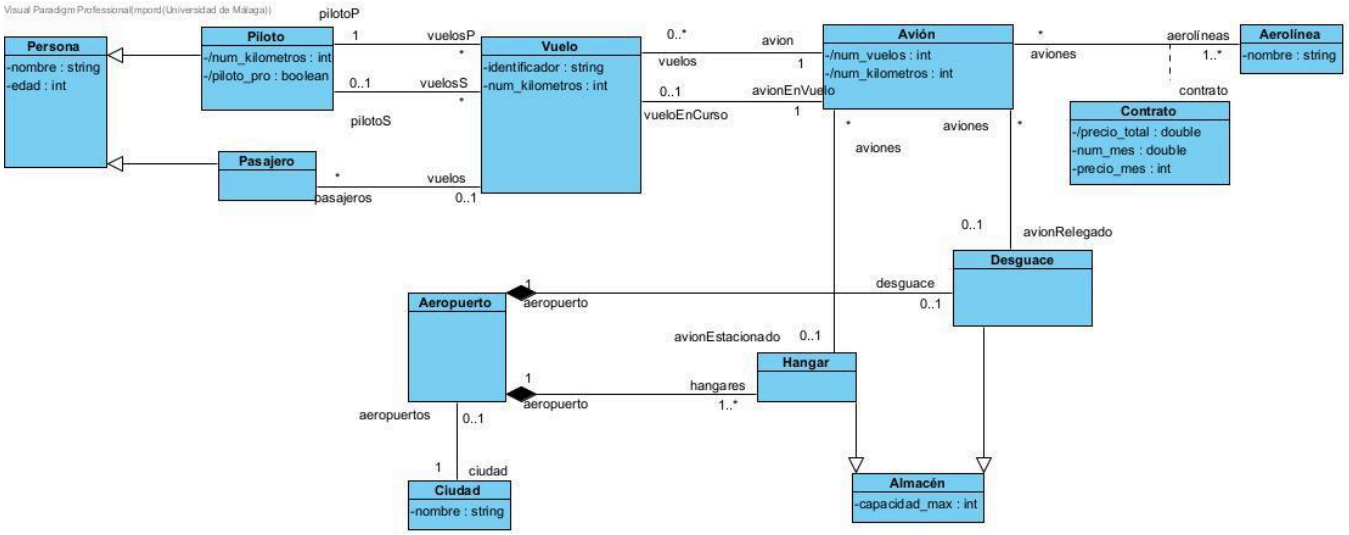
Introducción

En el siguiente documento se muestran los diagramas de clases en las aplicaciones Visual Paradigm y USE. Además de mostrar las restricciones a las que está sujeta el modelo, las variables derivadas que contiene y algunas notas aclaratorias para facilitar la comprensión y entendimiento del mismo, sus entidades, relaciones y diagramas.

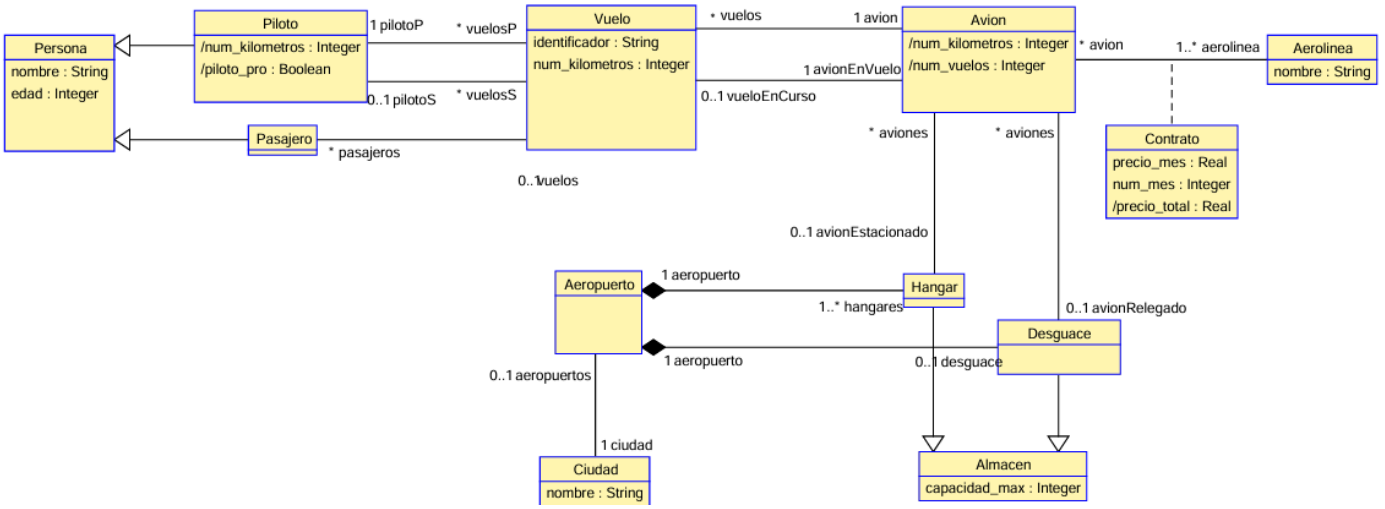
Al final del documento se recopilan los códigos de los diagramas y pruebas.

Diagramas de clases

- Visual Paradigm:



- USE:



Explicaciones adicionales

Herencias:

- "Piloto" y "Pasajero" heredan de persona: como hay que guardar el nombre y la edad de cada piloto y cada persona, hemos creado la clase padre "Persona".
- "Hangar" y "Desguace" heredan de "Almacén": como hay que guardar la capacidad máxima tanto de los hangares como de los desguaces, hemos creado la clase padre "Almacén"

Asociaciones:

- Estado del avión: un avión puede estar volando, estacionado o relegado, para especificar esto hemos optado por crear una relación de asociación 0..1 con "Avión" y "Hangar", "Desguace" y "Vuelo", por tanto un avión puede o no estar en un hangar, desguace o volando.
- "Piloto" y "Avión": estas dos relaciones permiten que para cada vuelo exista un piloto principal y uno o ningún piloto secundario. Además, los pilotos tanto principales como secundarios pueden participar en múltiples vuelos.
- "Pasajero" y "Avión": esta relación sirve para que el vuelo pueda guardar la información de todos los pasajeros que se encontraban.
- "Avión" y "Vuelo": encontramos aquí dos relaciones de asociación porque una, como se ha explicado anteriormente, es para saber si el avión está volando o no. Y la otra, en cambio, es para registrar qué avión se usó en cada vuelo y para que el avión guarde la información de todos los vuelos en los que ha volado.

Clases asociación:

- Hemos creado una clase asociación con "Avión" y "Aerolínea" para registrar los atributos del contrato que se genera cuando la aerolínea contrata un avión.

Composición:

- "Aeropuerto", "Hangar" y "Desguace": los hangares y desguaces están vinculados a un aeropuerto, por lo que, en caso de que este sea eliminado, todos los desguaces y hangares asociados deberán ser igualmente suprimidos.

Clases:

- Finalmente hemos añadido la clase "Ciudad" por si en el futuro es necesario incorporar más atributos.

Variables Derivadas

A continuación se realizarán las explicaciones referentes a los atributos derivados del modelo además de comprobaciones.

1. **“num_vuelos”** : indica la cantidad de vuelos realizados por un avión.

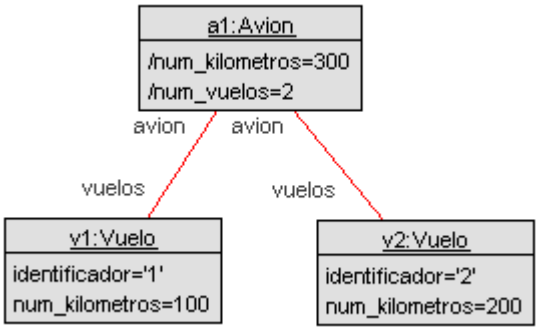
```
num_vuelos : Integer
            derive : self.vuelos -> size()
```

2. **“num_kilometros”(Clase Avión):** indica el número de kilómetros totales realizados por un avión.

```
num_kilometros : Integer
            derive : self.vuelos.num_kilometros -> sum()
```

Comprobación:

```
!new Avion('a1')


!new Vuelo('v1')
!v1.identificador := '1'
!v1.num_kilometros := 100


!new Vuelo('v2')
!v2.identificador := '2'
!v2.num_kilometros := 200


!insert (v1,a1) into vuelosAvion
!insert (v2,a1) into vuelosAvion
```



Como vemos, se calcula bien tanto el número de vuelos (viendo con cuántos vuelos está relacionado el avión) como el número de kilómetros (sumando el número de kilómetros de los vuelos con los que está relacionado un avión).

3. **“num_kilometros”(Clase Piloto):** indica el número de kilómetros totales realizados por un piloto.

```
num_kilometros : Integer
      derive :
         (self.vuelosP.num_kilometros) ->
union(self.vuelosS.num_kilometros) -> sum()
```

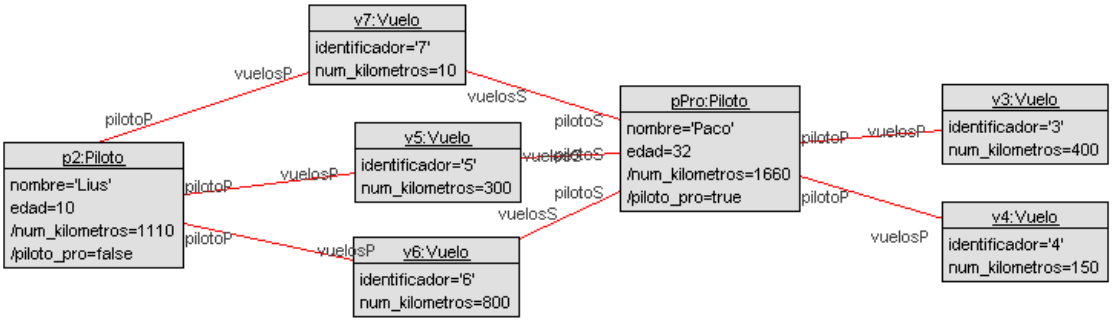
4. **“piloto_pro”** : indica si el piloto ha alcanzado el nivel de Pro.

```
piloto_pro : Boolean
       derive : (self.vuelosS ->size() >= 2000) and (self.vuelosP
-> size() >=1000)
```

Comprobación:

```
---------VUELOS----------
!new Vuelo('v3')
!v3.identificador := '3'
!v3.num_kilometros := 400

!new Vuelo('v4')
!v4.identificador := '4'
!v4.num_kilometros := 150

!new Vuelo('v5')
!v5.identificador := '5'
!v5.num_kilometros := 300

!new Vuelo('v6')
!v6.identificador := '6'
!v6.num_kilometros := 800

!new Vuelo('v7')
!v7.identificador := '7'
!v7.num_kilometros := 10
-------FIN VUELOS--------

---------PILOTOS---------
!new Piloto('pPro')
```

```
!pPro.nombre := 'Paco'
!pPro.edad := 32

!new Piloto('p2')
!p2.nombre := 'Lius'
!p2.edad := 10
-------FIN PILOTOS-------

!insert (v3,pPro) into vueloPilotoPrincipal
!insert (v4,pPro) into vueloPilotoPrincipal
!insert (v7,pPro) into vueloPilotoSecundario
!insert (v6,pPro) into vueloPilotoSecundario
!insert (v5,pPro) into vueloPilotoSecundario

!insert (v7,p2) into vueloPilotoPrincipal
!insert (v6,p2) into vueloPilotoPrincipal
!insert (v5,p2) into vueloPilotoPrincipal
```



Vemos que para ambos pilotos se calcula bien el número de kilómetros sumando los kilómetros de los vuelos que han realizado (tanto de principal como de secundario). También para el piloto 'Paco', se cambia el booleano de piloto_pro a true como se esperaría, en cambio Luis no lo tiene (para probar se ha establecido un mínimo de 3 vuelos como secundario y 2 como primario).

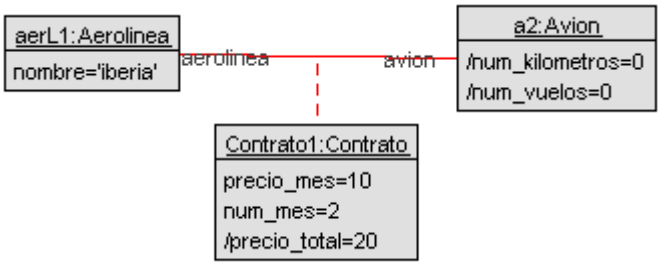
5. **"precio_total"**: indica el precio total que le cuesta a una aerolínea contratar un avión.

```
precio_total : Real
            derive : self.precio_mes * self.num_mes
```

Comprobación:

```
!new Aerolinea('aerL1')
!aerL1.nombre := 'iberia'

!new Avion ('a2')
```

```
!insert (a2,aerL1) into Contrato --Contrato1
!Contrato1.precio_mes := 10
!Contrato1.num_mes := 2
```



Vemos que al introducir los datos de los atributos precio_mes y num_mes, se autocalcula el precio_total del contrato.

Restricciones y Comprobaciones

En el siguiente apartado del documento se representan todas las invariantes/restricciones del modelo. En muchos casos los valores numéricos no coinciden con los que se piden, esto es con el objetivo de poder hacer las pruebas, pero en el código final esto estará cambiado:

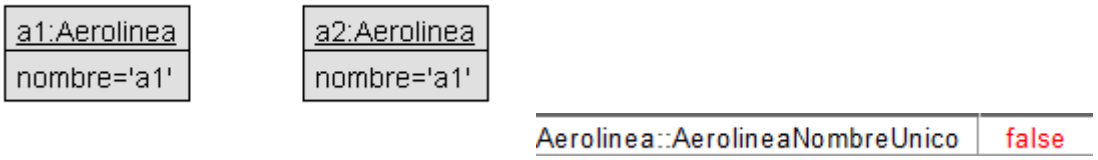
1. **"AerolineaNombreUnico"**: Todas las aerolíneas deben de tener distinto nombre.

- Restricción:

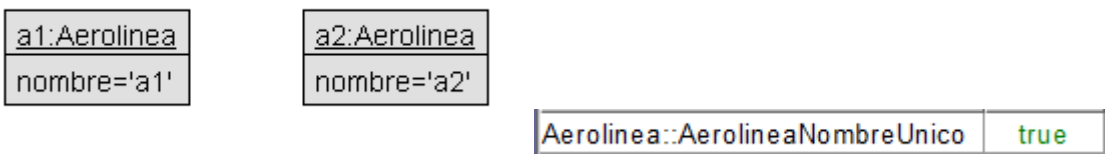
```
context Aerolinea
    inv AerolineaNombreUnico:
        Aerolinea.allInstances() -> forAll(a1, a2 | a1 <> a2 implies
a1.nombre <> a2.nombre)
```

- Comprobación:

```
!new Aerolinea('a1')
!new Aerolinea('a2')

!a1.nombre := 'a1'
!a2.nombre := 'a2'

check
```



En este primer caso las aerolíneas tienen el mismo nombre, por tanto la restricción no se cumple y devuelve false.



En este segundo caso vemos como el atributo nombre de la aerolínea a2 ha cambiado y ahora es distinto al de a1, por tanto la invariante se cumple y devuelve true.

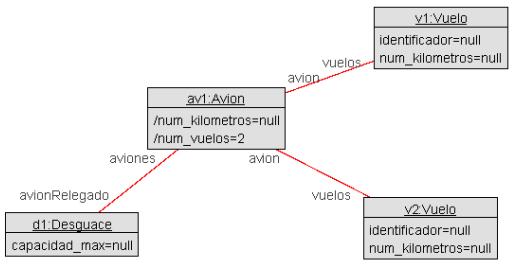
2. **"AvionDesguace"**: Un avión debe estar en el desguace si ha realizado más de 1.000 viajes.

- Restricción:

```
context Avion
    inv AvionDesguace:
        self.num_vuelos >= 1000 implies not
self.avionRelegado.oclIsUndefined()
        -- que el avión tenga más de 1000 vuelos implica que la
relación con desguace existe
```

- Comprobación:

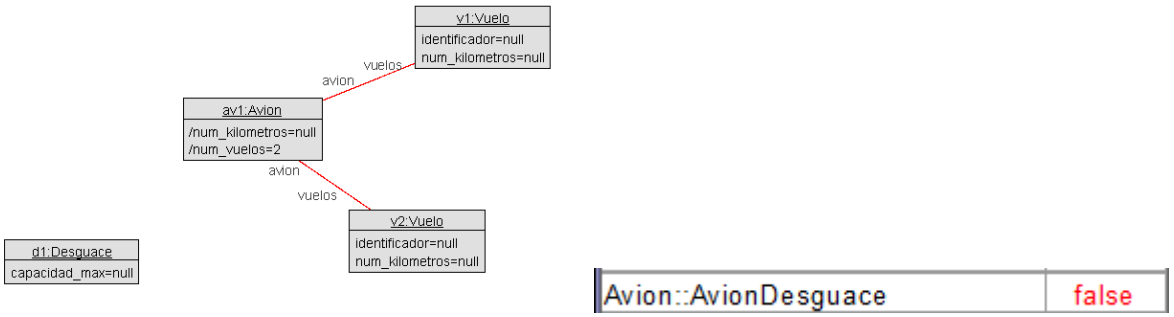
```
reset

!new Avion('av1')

!new Desguace('d1')

!new Vuelo('v1')
!new Vuelo('v2')

!insert (v1,av1) into vuelosAvion
!insert (v2,av1) into vuelosAvion

!insert (av1,d1) into avionDesguace--Descomentar esta línea de código
para comprobar que funciona
--Cambiar el límite de vuelos a 1 para que funcione

check
```



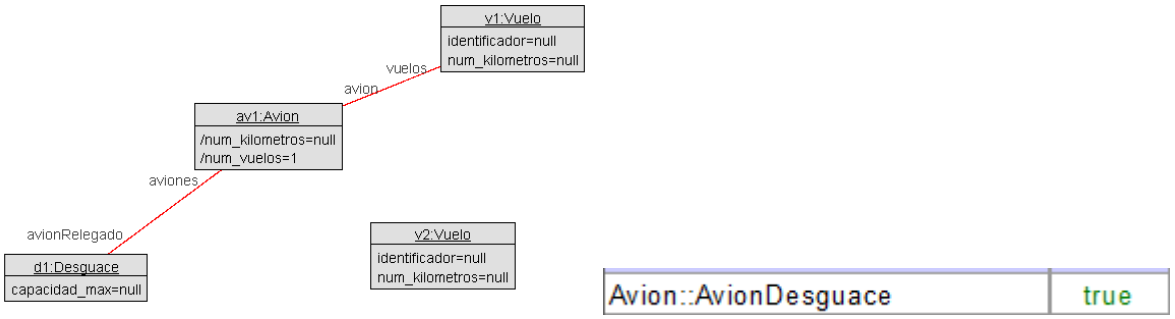
Avion::AvionDesguace	true

Aquí podemos observar la comprobación de la invariante en USE, en este primer caso como hemos establecido el límite de vuelos a 1 para que vaya al desguace y tenemos 2 vuelos,

debería haber una relación con desguace. Por tanto, en el .soil descomentamos la línea que crea dicha relación para que así el test devuelva true.



En este segundo caso, comentamos la línea de código anteriormente mencionada, como el avión se pasa del límite de vuelos pero no está en el desguace (la relación ya no existe), devuelve false.



Hemos supuesto que el avión puede estar en el desguace si tiene menos de 1000 vuelos porque puede haber tenido alguna rotura o algún fallo irreparable.

3. **“IdVueloUnico”**: El id de cada vuelo tiene que ser único.

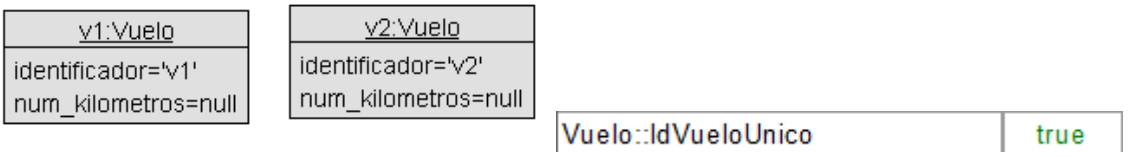
- Restricción:

```
context Vuelo
    inv IdVueloUnico:
        Vuelo.allInstances() -> forAll(v1,v2|  v1 <> v2 implies
v1.identificador <> v2.identificador)
```

- Comprobación:

```
reset

!new Vuelo('v1')
!new Vuelo('v2')

!v1.identificador := 'v1'
--Cambiar el identificador de v2 a v1 para que falle la invariante
!v2.identificador := 'v2'
```

```
check
```



En este primer caso, vemos que los vuelos al tener distinto identificador cumplen la restricción.

Y como es lógico, en el caso contrario de que los vuelos tengan distinto ID, obtenemos false.

4. **"PilotosDistintos"**: El piloto principal y el piloto secundario de un vuelo deben ser personas diferentes.

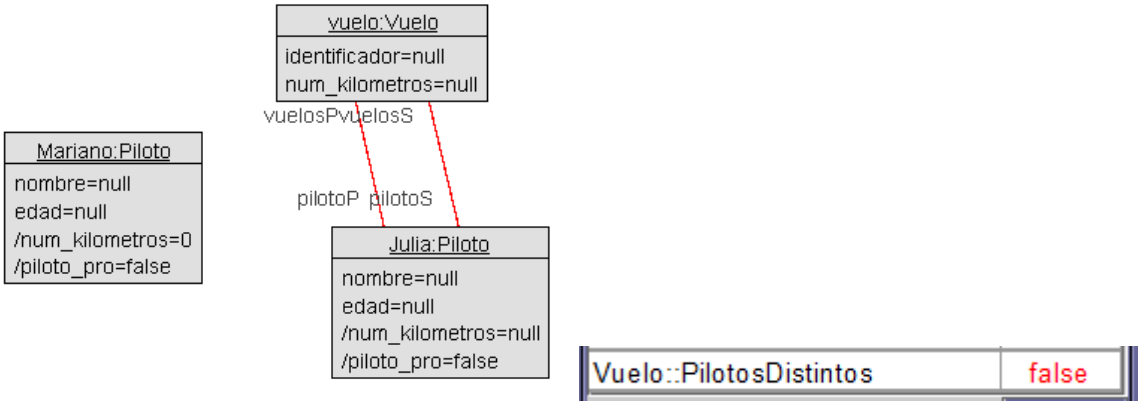
- Restricción:

```
context Vuelo
    inv PilotosDistintos:
        self.pilotoP <> self.pilotoS
```

- Comprobación:

```
!new Vuelo('vuelo')
!new Piloto('Julia')
!new Piloto('Mariano')


--Si se cambia Julia por Mariano en uno de los dos insert la invariante
--es correcta
!insert (vuelo, Julia) into vueloPilotoPrincipal
!insert (vuelo, Julia) into vueloPilotoSecundario
```

En este primer caso a un vuelo le asignamos dos pilotos, principal y secundario, pero los dos son el mismo piloto 'Julia', por lo que la invariante falla ya que deben ser distintos.



Sin embargo, cuando ahora cambiamos el piloto secundario por 'Mariano', la invariante es correcta

5. **"MaxAerolineasPorPiloto"**: Un piloto solo puede haber trabajado o trabajar en 2 o menos aerolíneas distintas.

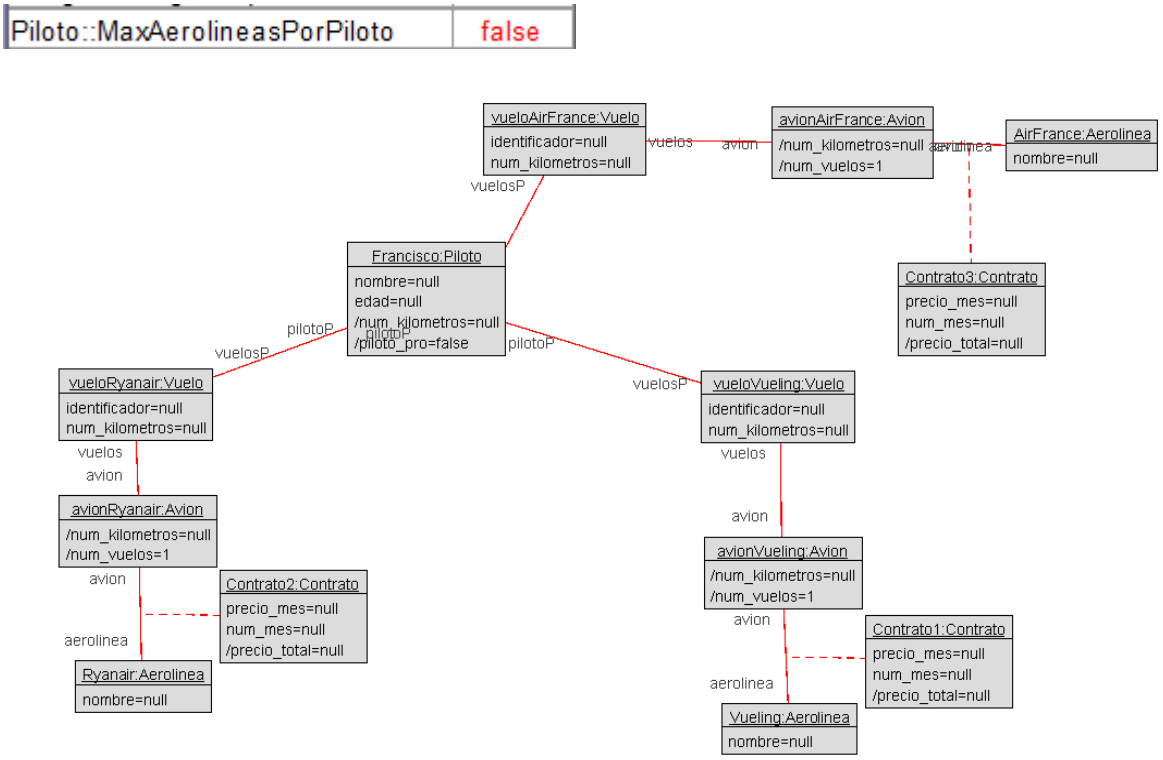
- Restricción:

```
context Piloto
    inv MaxAerolineasPorPiloto:
        self.vuelosP.avion.aerolinea ->
union(self.vuelosS.avion.aerolinea) -> asSet()->size()<3
```

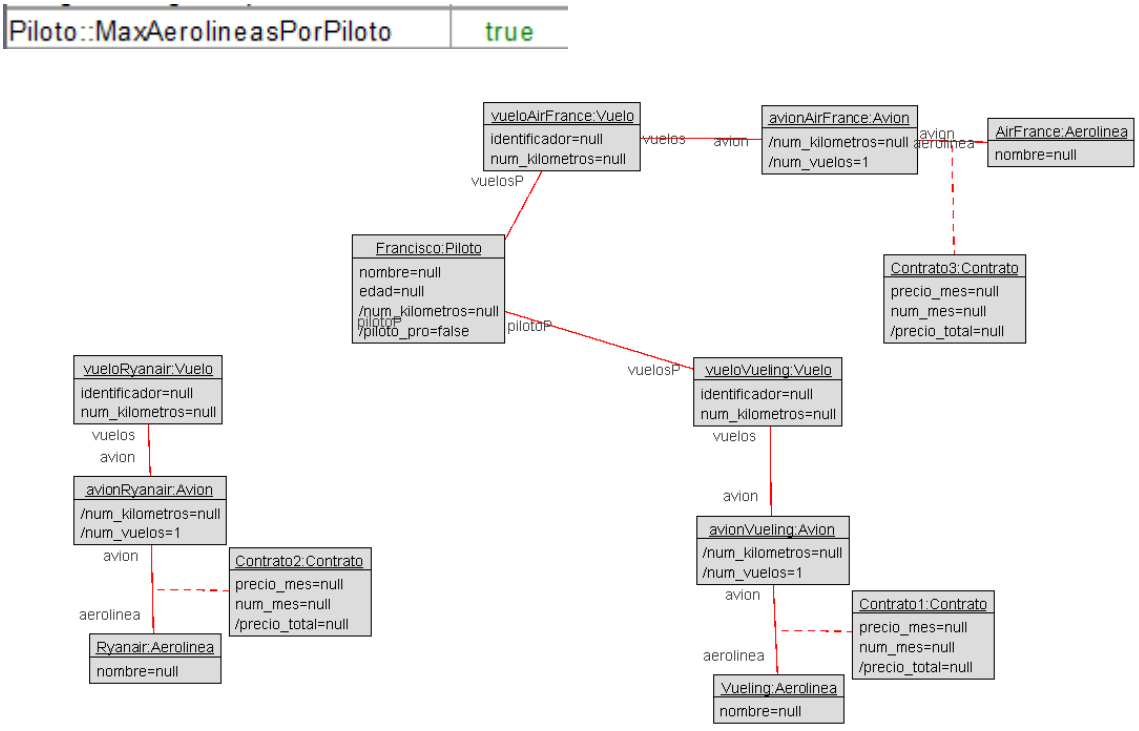
- Comprobación:

```
!new Aerolinea('Vueling')
!new Aerolinea('Ryanair')
!new Aerolinea('AirFrance')
```

```
!new Avion('avionVueling')
!new Avion('avionRyanair')
!new Avion('avionAirFrance')


!new Vuelo('vueloVueling')
!new Vuelo('vueloRyanair')
!new Vuelo('vueloAirFrance')


!new Piloto('Francisco')


!insert (vueloVueling, Francisco) into vueloPilotoPrincipal
!insert (vueloRyanair, Francisco) into vueloPilotoPrincipal
!insert (vueloAirFrance, Francisco) into vueloPilotoPrincipal

--Si comentamos una de estas relaciones de vuelos de diferentes
--aerolineas con el piloto Francisco, la restricción se cumplirá


!insert (vueloVueling, avionVueling) into vuelosAvion
!insert (vueloRyanair, avionRyanair) into vuelosAvion
!insert (vueloAirFrance, avionAirFrance) into vuelosAvion


!insert (avionVueling, Vueling) into Contrato
!insert (avionRyanair, Ryanair) into Contrato
!insert (avionAirFrance, AirFrance) into Contrato
```



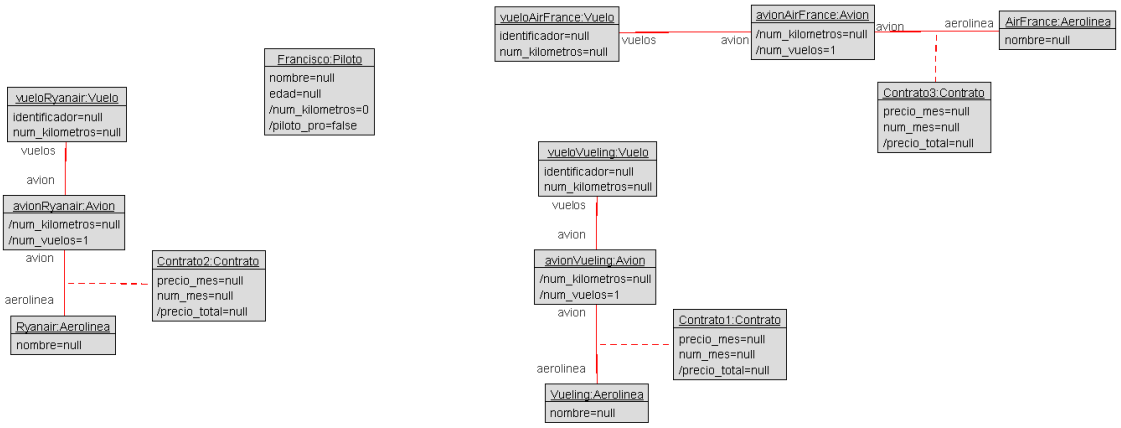
Para poder comprobar esta invariante, se ha creado un piloto, que está relacionado con tres vuelos, estos a su vez relacionados con diferentes aviones. Además, los aviones están relacionados cada uno con una aerolínea distinta. Por tanto se muestra que el piloto ha trabajado para más de dos aerolíneas y no se cumple la invariante.



Si comentamos una de las relaciones del piloto con uno de los vuelos, este solo habría trabajado en dos aerolíneas diferentes, entonces la restricción se cumpliría.

Nos aseguramos de que cuando un piloto no tiene asignada ninguna aerolínea, este invariante también funciona correctamente:

Piloto::MaxAerolineasPorPiloto	true



6. **"AvionEstadoObligatorio"**: Un avión debe de estar en uno de estos tres estados:
   a. avion_volando (Relación con vuelo)
   b. avion_estacionado (Relación con hangar)
   c. avion_relegado (Relación con desguace)

   Adicionalmente se comprueba que si un avión está en estado volando, ese vuelo debería estar registrado.

- Restricción:

```
context Avion
    inv AvionEstadoObligatorio:
        (self.vueloEnCurso -> size() + self.avionRelegado -> size() +
self.avionEstacionado -> size() = 1)
        and (not self.vueloEnCurso.oclIsUndefined() implies self.vuelos
-> includes(self.vueloEnCurso))
```

- Comprobación:

```
reset

!new Avion ('avion')

!new Vuelo('vuelo')
!new Hangar ('hangar')
!new Desguace('desguace')

!insert(vuelo, avion) into avionVolando
--!insert(avion, hangar) into avionHangar
--!insert(avion, desguace) into avionDesguace
```

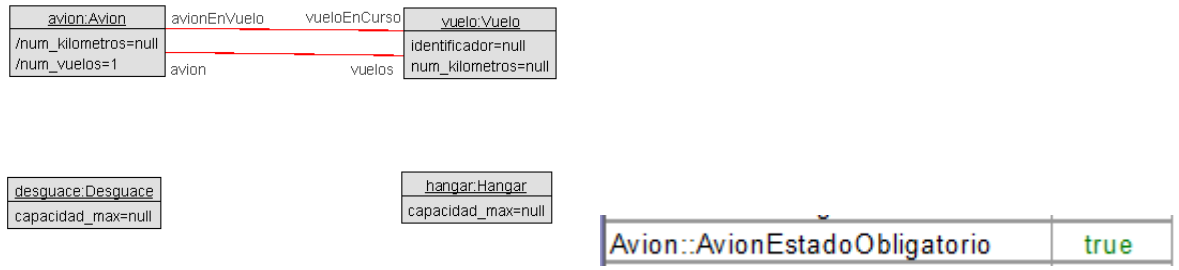
```
-- Si se descomenta una o ambas lineas, la invariante falla

!insert(vuelo, avion) into vuelosAvion

check
```

Si el avión tiene un estado, la invariante se cumple:

Pero al insertarlo en dos (o más) estados diferentes, se incumple la restricción:

Tampoco se cumple cuando un avión no tiene ningún estado:

Comprobamos que cuando el avión está en el estado volando, el vuelo debe haber sido registrado:



Si no está registrado, la invariante falla:

—----------EDU VA POR AQUI—----------

7. **“CapMaxHangar”**: Un avión no puede estar en un hangar si ya está completo

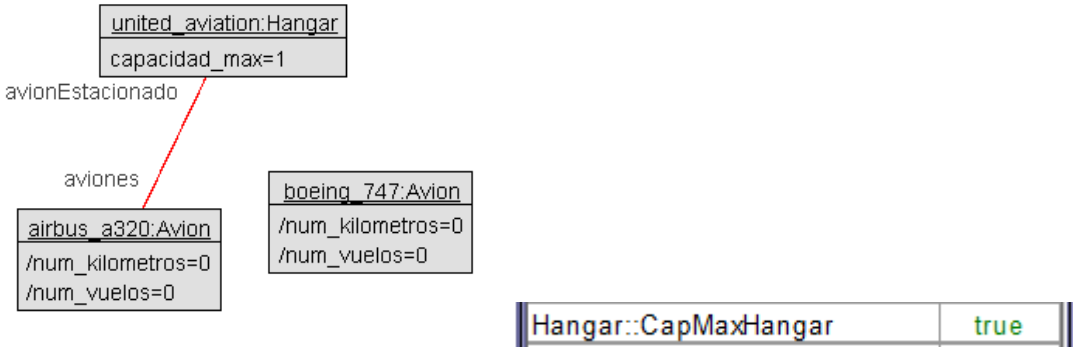
- Restricción:

```
context Hangar
    inv CapMaxHangar:
        self.aviones -> size() <= self.capacidad_max
```

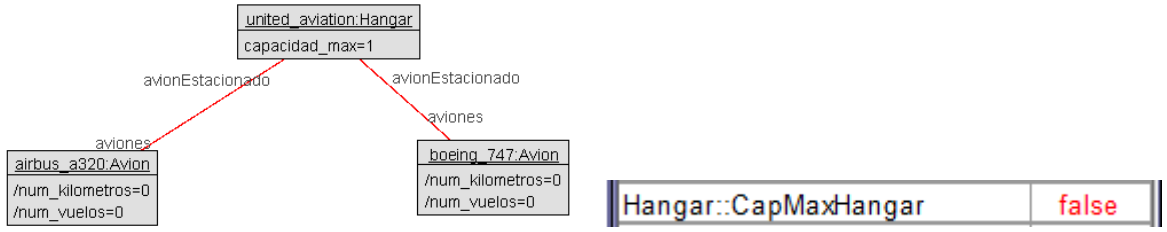
- Comprobación

```

--Comprobación invariante:"CapMaxHangar"

!new Avion('airbus_a320')
!new Avion('boeing_747')

!new Hangar('united_aviation')
!united_aviation.capacidad_max := 1

!insert(airbus_a320, united_aviation) into avionHangar
--Descomentar la linea de abajo para que falle
--Porque dos aviones estarían en un hangar de capacidad 1
--!insert(boeing_747, united_aviation) into avionHangar
```

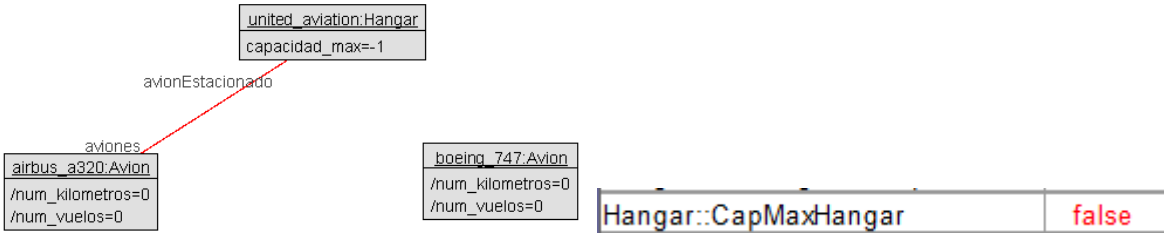
Cuando se inserta un avión en un hangar de capacidad máxima 1, se cumple la invariante.



Sin embargo, cuando insertamos otro avión, ya se supera su capacidad máxima y falla.



Para aumentar la robustez del sistema, comprobamos qué ocurre cuando la capacidad máxima del hangar es una cantidad negativa (capacidad_max := -1):



8. **“CapMaxDesguace”**: Un avión no puede estar en un desguace si ya está completo

- Restricción:

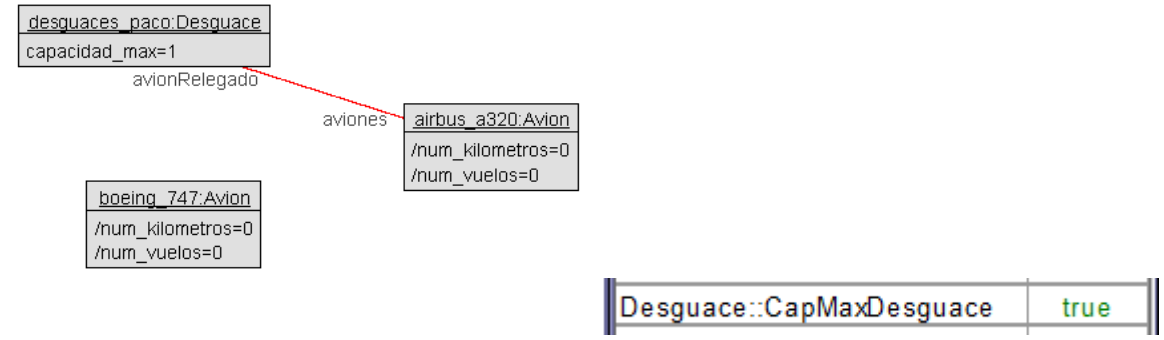
```
context Desguace
    inv CapMaxDesguace:
        self.aviones -> size() <= self.capacidad_max
```

- Comprobación

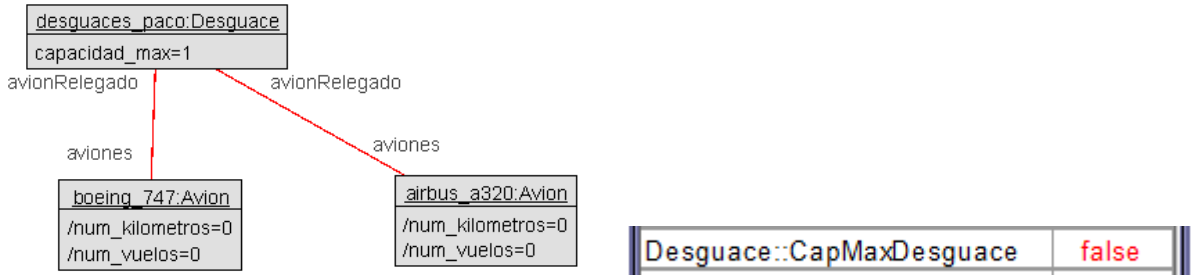
```
--Comprobación invariante:"CapMaxDesguace"

!new Avion('airbus_a320')
!new Avion('boeing_747')
```

```
!new Desguace('desguaces_paco')
!desguaces_paco.capacidad_max := 1

!insert(airbus_a320, desguaces_paco) into avionDesguace
--Descomentar la linea de abajo para que falle
--Porque dos aviones estarían en un desguace de capacidad 1
--!insert(boeing_747, desguaces_paco) into avionDesguace
```

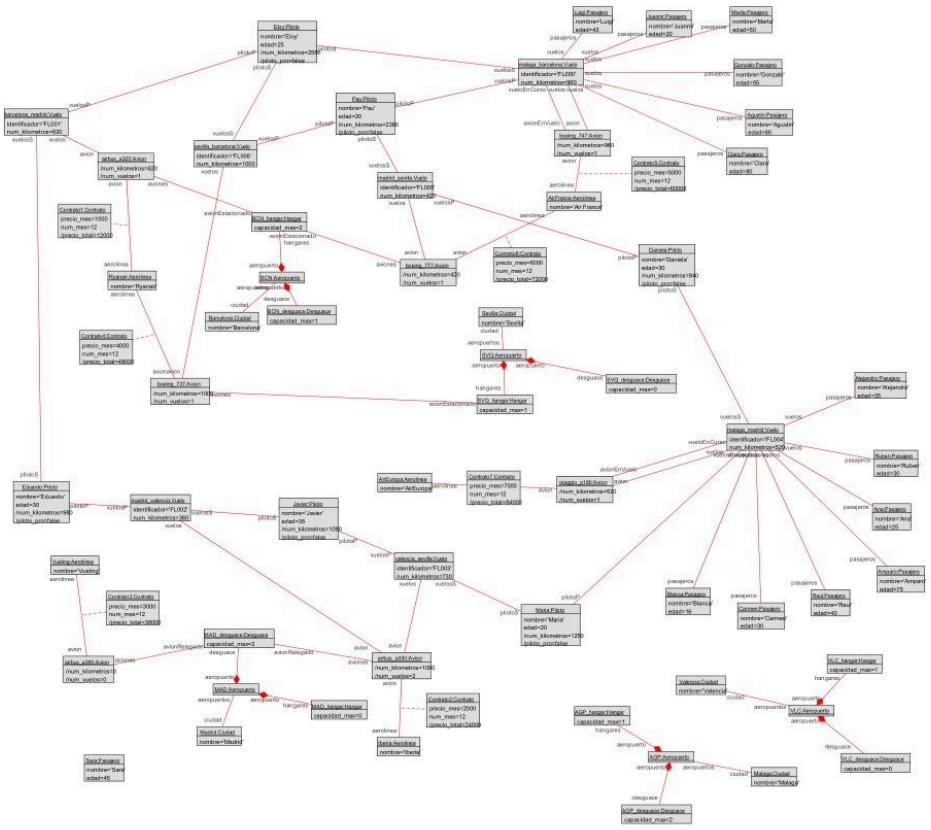
Cuando se inserta un avión en un desguace de capacidad máxima 1, se cumple la invariante.



Sin embargo, cuando insertamos otro avión, ya se supera su capacidad máxima y falla.



[DIAGRAMA DE OBJETOS MASIVO]



Por último, hemos creado un diagrama de objetos con un ejemplo real de lo que podría ser el sistema de aviación en funcionamiento.

Hemos creado varios vuelos, cada uno de ellos con su piloto principal (y secundario aunque es opcional), su avión con su respectiva aerolínea a la que pertenece, algunos con pasajeros, otros sin ellos, algunos se encuentran volando, otros en un desguace o hangar (que se encuentran en un aeropuerto que está en su respectiva ciudad), etc..

Class invariants

Invariant	Satisfied
Aerolinea::AerolineaNombreUnico	true
Avion::AvionDesguace	true
Avion::AvionEstadoUnico	true
Desguace::CapMaxDesguace	true
Hangar::CapMaxHangar	true
Piloto::MaxAerolineasPorPiloto	true
Vuelo::IdVueloUnico	true
Vuelo::PilotosDistintos	true

Explicit cnstrs. OK. Inherent cnstrs.... 100 %

Como se puede observar en la imagen, todas las invariantes se cumplen, por tanto como he mencionado anteriormente es un ejemplo válido de lo que el sistema podría recoger.

ACLARACIÓN:

Se ha comprobado que al hacer check en la consola para todos los .soil no se viola ninguna multiplicidad.

Códigos

Práctica1.use:

```
model Practica1

----------Clases-----------
class Avion
attributes
    num_kilometros : Integer
            derive : self.vuelos.num_kilometros -> sum()
    num_vuelos : Integer
            derive : self.vuelos -> size()
end

associationclass Contrato between
    Avion[0..*] role avion
    Aerolinea[1..*] role aerolinea
attributes
    precio_mes : Real
    num_mes : Integer
    precio_total : Real
            derive : self.precio_mes * self.num_mes
end

class Aerolinea
attributes
    nombre : String
end

class Aeropuerto
end

class Vuelo
attributes
    identificador : String
    num_kilometros : Integer
end

class Ciudad
attributes
    nombre : String
```

```
end

class Persona
attributes
    nombre : String
    edad : Integer
end

class Piloto < Persona
attributes
    num_kilometros : Integer derive :
        -- SE PUEDE HACER DE LAS DOS FORMAS

        --(self.vuelosP -> collect(v | v.num_kilometros))
        ---> union(self.vuelosS -> collect(v | v.num_kilometros))
        ---> sum()
            -- Se hace un collect de los num_kilometros de vuelosP y
vuelosS y se suman

        (self.vuelosP.num_kilometros) ->
union(self.vuelosS.num_kilometros) -> sum()
            -- Se hace una union de los kilometros de vuelosP y vuelosS
y se suman
    piloto_pro : Boolean
        derive : (self.vuelosS ->size() >= 2000) and (self.vuelosP ->
size() >=1000)
end

class Pasajero < Persona
end

class Almacen
attributes
    capacidad_max : Integer
end

class Hangar < Almacen
end

class Desguace < Almacen
end
```

```
---------Fin de Clases--------

----------Asociaciones---------
association vuelosAvion between
    Vuelo [0..*] role vuelos
    Avion [1] role avion
end

association avionVolando between
    Vuelo [0..1] role vueloEnCurso
    Avion [0..1] role avionEnVuelo
end

association vueloPilotoPrincipal between
    Vuelo [0..*] role vuelosP
    Piloto [1] role pilotoP
end

association vueloPilotoSecundario between
    Vuelo [0..*] role vuelosS
    Piloto [0..1] role pilotoS
end

association vueloPasajero between
    Vuelo [0..1] role vuelos
    Pasajero [0..*] role pasajeros
end

association aeropuertoCiudad between
    Aeropuerto [0..1] role aeropuertos
    Ciudad [1] role ciudad
end

association avionDesguace between
    Avion [0..*] role aviones
    Desguace [0..1] role avionRelegado
end

association avionHangar between
    Avion [0..*] role aviones
    Hangar [0..1] role avionEstacionado
end
```

```
----------Fin Asociaciones-----

----------Composiciones----------
composition aeropuertoHangar between
    Aeropuerto [1] role aeropuerto
    Hangar [1..*] role hangares
end

composition aeropuertoDesguace between
    Aeropuerto [1] role aeropuerto
    Desguace [0..1] role desguace
end
----------Fin composiciones----------

----------Invariantes----------
constraints

context Aerolinea
    inv AerolineaNombreUnico:
        Aerolinea.allInstances() -> forAll(a1, a2 | a1 <> a2 implies
a1.nombre <> a2.nombre)
                        -- el nombre de la aerolinea no puede ser nulo
y
                        -- si a1 es distinto de a2 entonces a1.nombre
es distinto de a2.nombre



context Vuelo
    inv IdVueloUnico:
        Vuelo.allInstances() -> forAll(v1,v2|  v1 <> v2 implies
v1.identificador <> v2.identificador)
                        -- el identificador del vuelo no puede ser nulo
y
                        -- si el objeto no es el mismo esto implica que
los identificadores son distintos

    inv PilotosDistintos:
        self.pilotoP <> self.pilotoS
            -- Si el piloto principal de mi vuelo es distinto del
secundario

context Avion
```

```
    inv AvionDesguace:
        self.num_vuelos >= 1000 implies not
self.avionRelegado.oclIsUndefined()
                        -- Si el numero de vuelos es mayor o igual a
1000 entonces el avion esta en desguace
                        -- es decir, la variable avionRelegado no es
nula

    inv AvionEstadoObligatorio:
        (self.vueloEnCurso -> size() + self.avionRelegado -> size() +
self.avionEstacionado -> size() = 1)
        and (not self.vueloEnCurso.oclIsUndefined() implies self.vuelos
-> includes(self.vueloEnCurso))
                -- Un avion tiene que estar en un estado
obligatoriamente
                -- Si un avion esta volando entonces tiene que estar
registrado en un vuelo

context Piloto
    inv MaxAerolineasPorPiloto:
        self.vuelosP.avion.aerolinea ->
union(self.vuelosS.avion.aerolinea) -> asSet()->size()<3
                -- Un piloto no puede volar en mas de 2 aerolineas

context Hangar
    inv CapMaxHangar:
        self.aviones -> size() <= self.capacidad_max
                -- La cantidad de aviones en un hangar no puede ser
mayor que la capacidad maxima del hangar

context Desguace
    inv CapMaxDesguace:
        self.aviones -> size() <= self.capacidad_max
                -- La cantidad de aviones en un desguace no puede ser
mayor que la capacidad maxima del desguace


----------Fin Invariantes------
```

Diagrama de Objetos: general.soil

```
reset

-----------------------------------Inicio de la creación de
objetos-----------------------------------

!new Avion('airbus_a320')
!new Avion('airbus_a350')
!new Avion('airbus_a380')
!new Avion('boeing_737')
!new Avion('boeing_747')
!new Avion('boeing_777')
!new Avion('piaggio_p180')

!new Aerolinea('AirEuropa')
!AirEuropa.nombre := 'AirEuropa'
!new Aerolinea('Iberia')
!Iberia.nombre := 'Iberia'
!new Aerolinea('Vueling')
!Vueling.nombre := 'Vueling'
!new Aerolinea('Ryanair')
!Ryanair.nombre := 'Ryanair'
!new Aerolinea('AirFrance')
!AirFrance.nombre := 'Air France'

!new Aeropuerto('AGP')
!new Aeropuerto('BCN')
!new Aeropuerto('MAD')
!new Aeropuerto('VLC')
!new Aeropuerto('SVQ')

!new Vuelo('malaga_barcelona')
!malaga_barcelona.identificador := 'FL000'
!malaga_barcelona.num_kilometros := 960
!new Vuelo('barcelona_madrid')
!barcelona_madrid.identificador := 'FL001'
!barcelona_madrid.num_kilometros := 620
!new Vuelo('madrid_valencia')
!madrid_valencia.identificador := 'FL002'
!madrid_valencia.num_kilometros := 360
!new Vuelo('valencia_sevilla')
!valencia_sevilla.identificador := 'FL003'
```

```
!valencia_sevilla.num_kilometros := 730
!new Vuelo('malaga_madrid')
!malaga_madrid.identificador := 'FL004'
!malaga_madrid.num_kilometros := 520
!new Vuelo('madrid_sevilla')
!madrid_sevilla.identificador := 'FL005'
!madrid_sevilla.num_kilometros := 420
!new Vuelo('sevilla_barcelona')
!sevilla_barcelona.identificador := 'FL006'
!sevilla_barcelona.num_kilometros := 1000

!new Ciudad('Malaga')
!Malaga.nombre := 'Malaga'
!new Ciudad('Barcelona')
!Barcelona.nombre := 'Barcelona'
!new Ciudad('Madrid')
!Madrid.nombre := 'Madrid'
!new Ciudad('Valencia')
!Valencia.nombre := 'Valencia'
!new Ciudad('Sevilla')
!Sevilla.nombre := 'Sevilla'

!new Piloto('Pau')
!Pau.nombre := 'Pau'
!Pau.edad := 20
!new Piloto('Eloy')
!Eloy.nombre := 'Eloy'
!Eloy.edad := 25
!new Piloto('Eduardo')
!Eduardo.nombre := 'Eduardo'
!Eduardo.edad := 30
!new Piloto('Javier')
!Javier.nombre := 'Javier'
!Javier.edad := 35
!new Piloto('Maria')
!Maria.nombre := 'Maria'
!Maria.edad := 20
!new Piloto('Daniela')
!Daniela.nombre := 'Daniela'
!Daniela.edad := 30

!new Pasajero('Gonzalo')
!Gonzalo.nombre := 'Gonzalo'
```

```
!Gonzalo.edad := 55
!new Pasajero('Agustin')
!Agustin.nombre := 'Agustin'
!Agustin.edad := 60
!new Pasajero('Juanmi')
!Juanmi.nombre := 'Juanmi'
!Juanmi.edad := 20
!new Pasajero('Clara')
!Clara.nombre := 'Clara'
!Clara.edad := 80
!new Pasajero('Ana')
!Ana.nombre := 'Ana'
!Ana.edad := 25
!new Pasajero('Ruben')
!Ruben.nombre := 'Ruben'
!Ruben.edad := 30
!new Pasajero('Alejandro')
!Alejandro.nombre := 'Alejandro'
!Alejandro.edad := 35
!new Pasajero('Amparo')
!Amparo.nombre := 'Amparo'
!Amparo.edad := 75
!new Pasajero('Blanca')
!Blanca.nombre := 'Blanca'
!Blanca.edad := 16
!new Pasajero('Luigi')
!Luigi.nombre := 'Luigi'
!Luigi.edad := 43
!new Pasajero('Marta')
!Marta.nombre := 'Marta'
!Marta.edad := 50
!new Pasajero('Sara')
!Sara.nombre := 'Sara'
!Sara.edad := 45
!new Pasajero('Raul')
!Raul.nombre := 'Raul'
!Raul.edad := 40
!new Pasajero('Carmen')
!Carmen.nombre := 'Carmen'
!Carmen.edad := 30

!new Hangar('AGP_hangar')
!AGP_hangar.capacidad_max := 1
```

```
!new Hangar('BCN_hangar')
!BCN_hangar.capacidad_max := 2
!new Hangar('MAD_hangar')
!MAD_hangar.capacidad_max := 0
!new Hangar('VLC_hangar')
!VLC_hangar.capacidad_max := 1
!new Hangar('SVQ_hangar')
!SVQ_hangar.capacidad_max := 1

!new Desguace('AGP_desguace')
!AGP_desguace.capacidad_max := 2
!new Desguace('BCN_desguace')
!BCN_desguace.capacidad_max := 1
!new Desguace('MAD_desguace')
!MAD_desguace.capacidad_max := 2
!new Desguace('VLC_desguace')
!VLC_desguace.capacidad_max := 0
!new Desguace('SVQ_desguace')
!SVQ_desguace.capacidad_max := 0

----------------------------------Fin de la creación de
objetos-----------------------------------

----------------------------------Inicio de las
asociaciones-----------------------------------

!insert(airbus_a320, Ryanair) into Contrato --Contrato1
!Contrato1.precio_mes:= 1000
!Contrato1.num_mes:= 12
!insert(airbus_a350, Iberia) into Contrato --Contrato2
!Contrato2.precio_mes:= 2000
!Contrato2.num_mes:= 12
!insert(airbus_a380, Vueling) into Contrato --Contrato3
!Contrato3.precio_mes:= 3000
!Contrato3.num_mes:= 12
!insert(boeing_737, Ryanair) into Contrato --Contrato4
!Contrato4.precio_mes:= 4000
!Contrato4.num_mes:= 12
!insert(boeing_747, AirFrance) into Contrato --Contrato5
!Contrato5.precio_mes:= 5000
!Contrato5.num_mes:= 12
!insert(boeing_777, AirFrance) into Contrato --Contrato6
!Contrato6.precio_mes:= 6000
```

```
!Contrato6.num_mes:= 12
!insert(piaggio_p180, AirEuropa) into Contrato --Contrato7
!Contrato7.precio_mes:= 7000
!Contrato7.num_mes:= 12

--Hay que poner las dos asociaciones? We are not that sure

!insert(malaga_barcelona, boeing_747) into vuelosAvion
!insert(barcelona_madrid, airbus_a320) into vuelosAvion
!insert(madrid_valencia, airbus_a350) into vuelosAvion
!insert(valencia_sevilla, airbus_a350) into vuelosAvion
!insert(malaga_madrid, piaggio_p180) into vuelosAvion
!insert(madrid_sevilla, boeing_777) into vuelosAvion
!insert(sevilla_barcelona, boeing_737) into vuelosAvion

!insert(malaga_barcelona, boeing_747) into avionVolando
!insert(malaga_madrid, piaggio_p180) into avionVolando

!insert(airbus_a320, BCN_hangar) into avionHangar
!insert(boeing_737, SVQ_hangar) into avionHangar
!insert(boeing_777, BCN_hangar) into avionHangar

!insert(airbus_a350, MAD_desguace) into avionDesguace
!insert(airbus_a380, MAD_desguace) into avionDesguace

!insert(AGP, Malaga) into aeropuertoCiudad
!insert(BCN, Barcelona) into aeropuertoCiudad
!insert(MAD, Madrid) into aeropuertoCiudad
!insert(VLC, Valencia) into aeropuertoCiudad
!insert(SVQ, Sevilla) into aeropuertoCiudad

!insert(AGP, AGP_hangar) into aeropuertoHangar
!insert(BCN, BCN_hangar) into aeropuertoHangar
!insert(MAD, MAD_hangar) into aeropuertoHangar
!insert(VLC, VLC_hangar) into aeropuertoHangar
!insert(SVQ, SVQ_hangar) into aeropuertoHangar

!insert(AGP, AGP_desguace) into aeropuertoDesguace
!insert(BCN, BCN_desguace) into aeropuertoDesguace
!insert(MAD, MAD_desguace) into aeropuertoDesguace
!insert(VLC, VLC_desguace) into aeropuertoDesguace
!insert(SVQ, SVQ_desguace) into aeropuertoDesguace
```

```
!insert(malaga_barcelona, Pau) into vueloPilotoPrincipal
!insert(barcelona_madrid, Eloy) into vueloPilotoPrincipal
!insert(madrid_valencia, Eduardo) into vueloPilotoPrincipal
!insert(valencia_sevilla, Javier) into vueloPilotoPrincipal
!insert(malaga_madrid, Maria) into vueloPilotoPrincipal
!insert(madrid_sevilla, Daniela) into vueloPilotoPrincipal
!insert(sevilla_barcelona, Pau) into vueloPilotoPrincipal

!insert(malaga_barcelona, Eloy) into vueloPilotoSecundario
!insert(barcelona_madrid, Eduardo) into vueloPilotoSecundario
!insert(madrid_valencia, Javier) into vueloPilotoSecundario
!insert(valencia_sevilla, Maria) into vueloPilotoSecundario
!insert(malaga_madrid, Daniela) into vueloPilotoSecundario
!insert(madrid_sevilla, Pau) into vueloPilotoSecundario
!insert(sevilla_barcelona, Eloy) into vueloPilotoSecundario

!insert(malaga_barcelona, Gonzalo) into vueloPasajero
!insert(malaga_barcelona, Agustin) into vueloPasajero
!insert(malaga_barcelona, Juanmi) into vueloPasajero
!insert(malaga_barcelona, Clara) into vueloPasajero
!insert(malaga_barcelona, Luigi) into vueloPasajero
!insert(malaga_barcelona, Marta) into vueloPasajero

!insert(malaga_madrid, Ana) into vueloPasajero
!insert(malaga_madrid, Ruben) into vueloPasajero
!insert(malaga_madrid, Alejandro) into vueloPasajero
!insert(malaga_madrid, Amparo) into vueloPasajero
!insert(malaga_madrid, Blanca) into vueloPasajero
!insert(malaga_madrid, Carmen) into vueloPasajero
!insert(malaga_madrid, Raul) into vueloPasajero
```