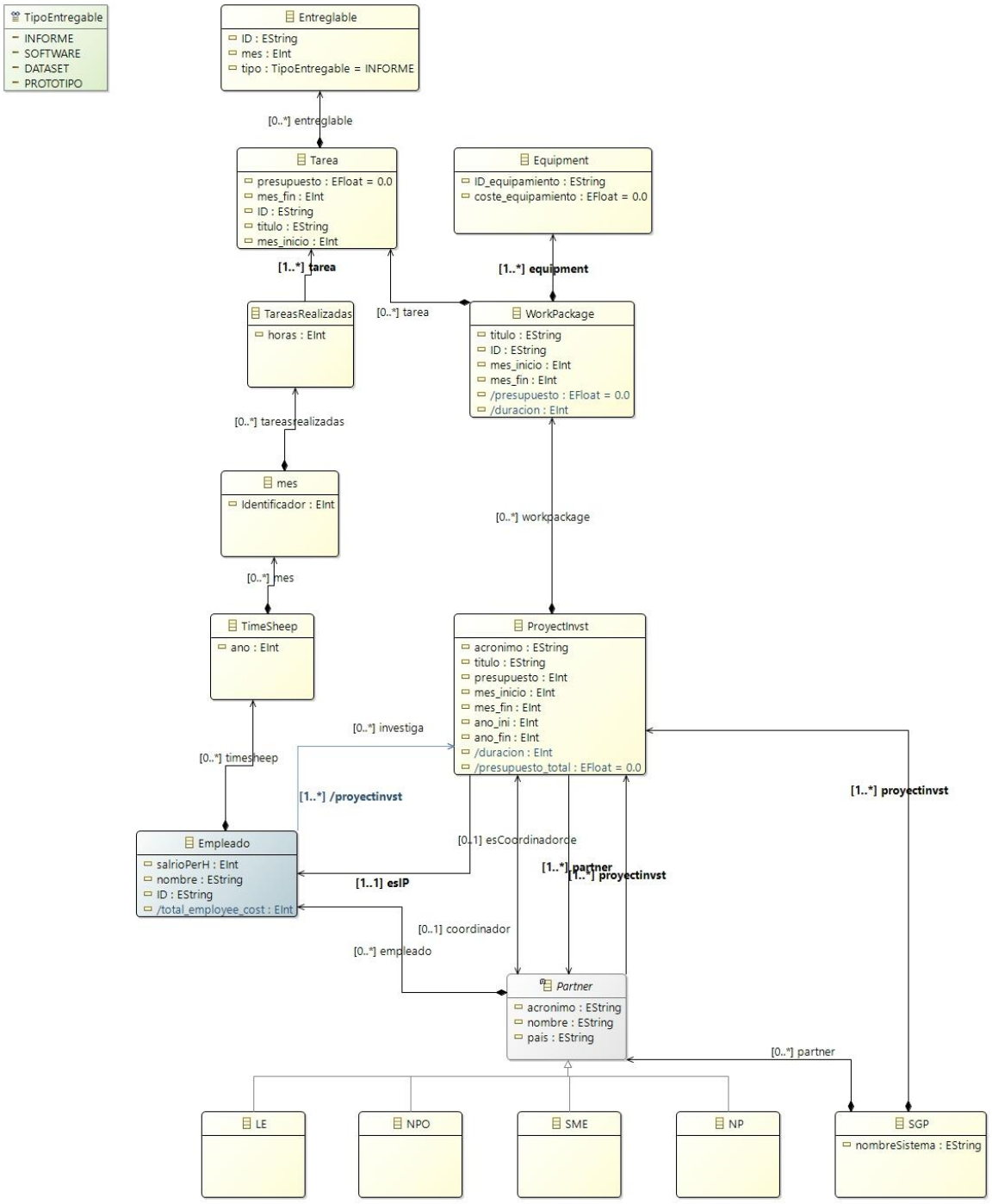
Actividad 2. Utilizando OCL para enriquecer el metamodelo de Gestión de Proyectos de Investigación y consultar algunos modelos de ejemplo

A. Figura del metamodelo



B. Ejercicio 1 Atributos derivados

- D01. Para todos los proyectos, paquetes de trabajo (WP) y tareas, se debe añadir un atributo derivado para obtener su duración total en meses

Context ProyectInvst

```
attribute duracion : ecore::EInt[?] { derived transient volatile }
             {
             initial:
             let
              meses_proyecto : Integer = (self.ano_fin - self.ano_ini) * 12 +
             (self.mes_fin - self.mes_inicio)
            in
             meses_proyecto;
             }
```

Context WorkPackage

```
attribute duracion : ecore::EInt[?] { derived transient volatile }
                                        {
                                               initial:
                    self.mes_fin - self.mes_inicio;
                                                    }
```

- D02. Calcula el presupuesto de un proyecto de investigación a partir de los presupuestos de sus paquetes de trabajo. (0.25 puntos)

Context ProyectInvst

```
attribute presupuesto_total : ecore::EFloat[?] { derived transient volatile }
                    {
                           initial:
          self.workpackage->iterate(wp : WorkPackage; total : ecore::EFloat =
0.0 | total + wp.presupuesto);
                                  }
```

D3. En empleado, añade un atributo denominado total_employee_cost donde se calcule el coste total del empleado teniendo en cuenta todos los proyectos en los que tiene horas asignadas a tareas. (0.5 puntos)

Context Empleado

```
attribute total_employee_cost : ecore::EInt[?] { derived transient volatile }
             {
             initial:
             let totalHoras : ecore::EInt =
             self.timesheep.mes.tareasrealizadas->iterate( tr :
TareasRealizadas; horastotales : ecore::EInt=0 | horastotales + tr.horas )
                                 in   totalHoras * self.salrioPerH;
                                 }
```

D4. En empleado, añade una referencia researchprojects que relacione cada empleado con todos los proyectos en los que tienen tareas asignadas. (0.5 puntos)

Context Empleado

property researchprojects : *ProyectInvst*[+**|1**] { **ordered derived** };

D5. El presupuesto de los paquetes de trabajo debe calcularse considerando tanto el equipamiento del paquete de trabajo como el coste de las horas de trabajo asignadas a los distintos trabajadores. (0.75 puntos)

Context WorkPackage

```
attribute presupuesto : ecore::EFloat[?] { derived transient volatile }
                                    {
                                          initial:
                                          let
                                          coste_trabajadores : ecore::EFloat =0.0,
                                          coste_equipo : ecore::EFloat =
self.equipment->collect(equip : Equipment | equip.coste_equipamiento)->sum()

                                          in
                                           coste_trabajadores+coste_equipo;
                                          }
```

INVARIANTES

I1. El mes de comienzo y conclusión y finalización de los proyectos debe estar comprendido entre [1,12] (0,25 puntos).

Context ProyectInvst

```
Invariant mes_correcto:
(self.mes_inicio <= 12 and self.mes_inicio > 0) and (self.mes_fin <= 12 and
self.mes_fin > 0);
```

I2. El número de un mes debe tener un valor en el rango [1,12] (0,25 puntos).

Context MES

```
invariant num_mes:
self.Identificador>0 and self.Identificador<12;
```

i3.En una hoja de horas, todos los meses deben tener distinto número de mes. (0,5 puntos).

Context TimeSheep

```
Invariant meses_distintos:
self.mes->forAll(id1, id2 |id1<>id2 implies
id1.Identificador<>id2.Identificador);
```

I4. El líder o coordinador del proyecto debe ser empleado de la organización que coordina el proyecto (0,5 puntos).

Context ProyectInvst

```
invariant empleado_coordinador:
self.coordinador.empleado->includes(self.esIP);
```

I5. Un empleado puede trabajar un máximo de 143 horas al mes, por tanto, la suma de horas asignadas a tareas en distintos proyectos no puede superar las 143 horas (0,5 puntos).

Context Empleado

```
Invariant horas_maximas:
	let
	horas_limite : ecore::EInt=143,
	horas_reales :ecore::EInt = self.timesheep.mes.tareasrealizadas-
	>iterate(tarea: TareasRealizadas; horas : ecore::EInt = 0 | horas +
	tarea.horas)
	in
	horas_limite>=horas_reales;
```

I6. Un proyecto no puede concluir antes de empezar (0,5 puntos).

Context ProyectInvst

```
invariant concluir:
self.duracion>0;
```

C. Ejercicio 3 (máximo 2 puntos)

Resultados

Consulta Nº.	Contexto	Texto de la consulta	Resultados de la consulta
Q1	WorkPackage Marketing	`self.tarea->size()`	`3`
Q2	ProyectInvst Onmi	`self.workpackage.tarea-> size()`	`3`
Q3	Empleado Juan Pablo	`self.timesheep.mes.tareasrealizadas.tarea.ID`	`'T2'` `'t3'`
Q4	WorkPackage Marketing	`self.workpackage.tarea.entregable- >select(tipo=TipoEntregable::DATASET)`	`Entreglable E7` `Entreglable E8`
Q5	SGP Sistema de gestion	`self.partner- >select(proyectinvst- >notEmpty()).acronimo`	`'AC'` `'AMW'` `'UEX'`
Q6	SGP Sistema de gestion	`self.partner->select(p | p.oclIsKindOf(NPO) and p.esCoordinadorde- >notEmpty()).acronimo`	`'UEX'`

D.CODIGO ECORE

```
import ecore : 'http://www.eclipse.org/emf/2002/Ecore' ;

package dmss_pa02 : dmss_pa02 = 'http://www.example.org/dmss_pa02'
{
	class SGP
	{
		property proyectinvst : ProyectInvst[+|1] { ordered composes };
		attribute nombreSistema : String[?];
		property partner : Partner[*|1] { ordered composes };
	}
	abstract class Partner
	{
		property proyectinvst : ProyectInvst[+|1] { ordered };
		property esCoordinadorde#coordinador : ProyectInvst[?];
		attribute acronimo : String[?];
		attribute nombre : String[?];
		attribute pais : String[?];
		property empleado : Empleado[*|1] { ordered composes };
	}
	class ProyectInvst
	{
		property partner : Partner[+|1] { ordered };
		property workpackage : WorkPackage[*|1] { ordered composes };
		attribute acronimo : String[?];
		attribute titulo : String[?];
		attribute presupuesto : ecore::EInt[?];
		attribute mes_inicio : ecore::EInt[?];
		attribute mes_fin : ecore::EInt[?];
		attribute ano_ini : ecore::EInt[?];
		attribute ano_fin : ecore::EInt[?];
		property esIP : Empleado[1];
		property coordinador#esCoordinadorde : Partner[?];
		attribute duracion : ecore::EInt[?] { derived transient volatile }
		{
			initial:
			let
		meses_proyecto : Integer = (self.ano_fin - self.ano_ini) * 12 +
(self.mes_fin - self.mes_inicio)
		    in
		     meses_proyecto;
			}
			attribute presupuesto_total : ecore::EFloat[?] { derived
transient volatile }
			{
				initial:
	    self.workpackage->iterate(wp : WorkPackage; total : ecore::EFloat =
0.0 | total + wp.presupuesto);
				}
				invariant
				mes_correcto:
				(self.mes_inicio <= 12 and self.mes_inicio > 0) and
(self.mes_fin <= 12 and self.mes_fin > 0);
				invariant empleado_coordinador:
				self.coordinador.empleado->includes(self.esIP);

				invariant concluir:
				self.duracion>0;
			}
			class Empleado
			{
				annotation _'http://www.obeo.fr/dsl/dnc/archetype'
				(
					archetype = 'Description'
				);
```

```
                property timesheep : TimeSheep[*|1] { ordered
composes };
                attribute salrioPerH : ecore::EInt[?];
                attribute nombre : String[?];
                attribute ID : String[?];
                property investiga : ProyectInvst[*|1] { ordered };
                attribute total_employee_cost : ecore::EInt[?] {
derived transient volatile }
                {
                        initial:
                        let totalHoras : ecore::EInt =
                        self.timesheep.mes.tareasrealizadas->iterate(
tr : TareasRealizadas; horastotales : ecore::EInt=0 | horastotales + tr.horas )
                        in   totalHoras * self.salrioPerH;
                        }
                        property proyectinvst : ProyectInvst[+|1] {
ordered derived };
                        invariant
                        horas_maximas:
                        let
                        horas_limite : ecore::EInt=143,
                        horas_reales :ecore::EInt =
self.timesheep.mes.tareasrealizadas->iterate(tarea: TareasRealizadas; horas :
ecore::EInt = 0 | horas + tarea.horas)
                        in
                        horas_limite>=horas_reales;
                }
                class WorkPackage
                {
                        property tarea : Tarea[*|1] { ordered composes
};
                        attribute titulo : String[?];
                        attribute ID : String[?];
                        attribute mes_inicio : ecore::EInt[?];
                        attribute mes_fin : ecore::EInt[?];
                        property equipment : Equipment[+|1] { ordered
composes };
                        attribute presupuesto : ecore::EFloat[?] {
derived transient volatile }
                        {
                                initial:
                                let
                                coste_trabajadores : ecore::EFloat =0.0,
                                coste_equipo : ecore::EFloat =
self.equipment->collect(equip : Equipment | equip.coste_equipamiento)->sum()

                                in
                                 coste_trabajadores+coste_equipo;
                                }
                                attribute duracion : ecore::EInt[?] {
derived transient volatile }
                                {
                                        initial:
                    self.mes_fin - self.mes_inicio;
                                        }

                                        invariant mes_comienzo:
                                         self.mes_inicio>0 and
mes_inicio<mes_fin;
                                         invariant mes_fin:
                                         self.mes_fin>0 and
mes_fin<duracion;


                                }
```

```
class Tarea
{
	property entreglable :
Entreglable[*|1] { ordered composes };
	attribute presupuesto :
ecore::EFloat[?];
	attribute mes_fin :
ecore::EInt[?];
	attribute ID : String[?];
	attribute titulo : String[?];
	attribute mes_inicio :
ecore::EInt[?];
}
class Entreglable
{
	attribute ID : String[?];
	attribute mes : ecore::EInt[?];
	attribute tipo :
TipoEntregable[?];
}
class SME extends Partner;
class NPO extends Partner;
class LE extends Partner;
class NP extends Partner;
class TimeSheep
{
	property mes : mes[*|1] { ordered
composes };
	attribute ano : ecore::EInt[?];
	invariant
	meses_distintos:
	self.mes->forAll(id1, id2
|id1<>id2 implies id1.Identificador<>id2.Identificador);
}
enum TipoEntregable { serializable }
{
	literal INFORME;
	literal SOFTWARE = 1;
	literal DATASET = 2;
	literal PROTOTIPO = 3;
}
class mes
{
	attribute Identificador :
ecore::EInt[?];
	property tareasrealizadas :
TareasRealizadas[*|1] { ordered composes };
	invariant num_mes:
	self.Identificador>0 and
self.Identificador<12;

}
class TareasRealizadas
{
	property tarea : Tarea[+|1] {
ordered };
	attribute horas : ecore::EInt[?];
}
class Equipment
{
	attribute ID_equipamiento :
String[?];
	attribute coste_equipamiento :
ecore::EFloat[?];
}
```

```
}
```