OCL Scribe

OCL Scribe

Integrating Modelio and USEOCL

oussama housni

<add your name here>

jean-marie favre

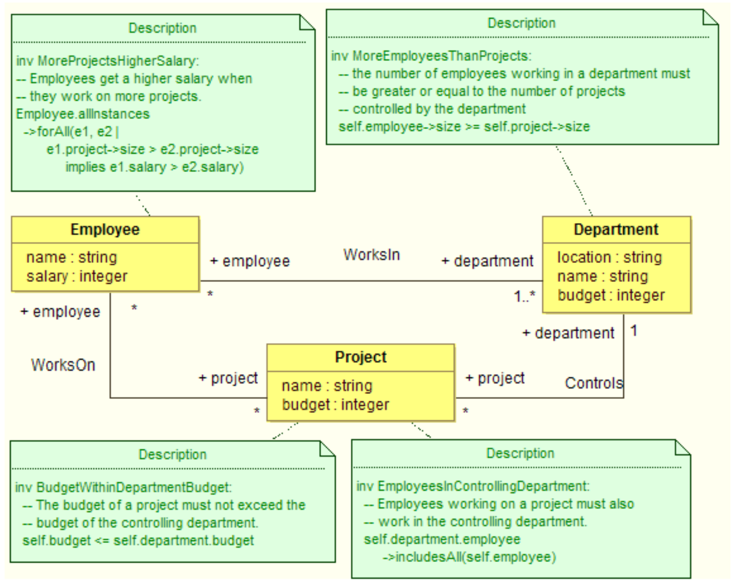
part of MODELIO Scribes

on open source project on github SOCIAL CODING

References

MODELIO Scribes	github.com/megaplan/ModelioScribes
MODELIO	modelio.org
USE OCL	sourceforge.net/projects/useocl

OCLScribe In a Nutshell



Description

```
inv MoreProjectsHigherSalary:
-- Employees get a higher salary when
-- they work on more projects.
Employee.allInstances
  ->forAll(e1, e2 |
      e1.project->size > e2.project->size
         implies e1.salary > e2.salary)
```

Description

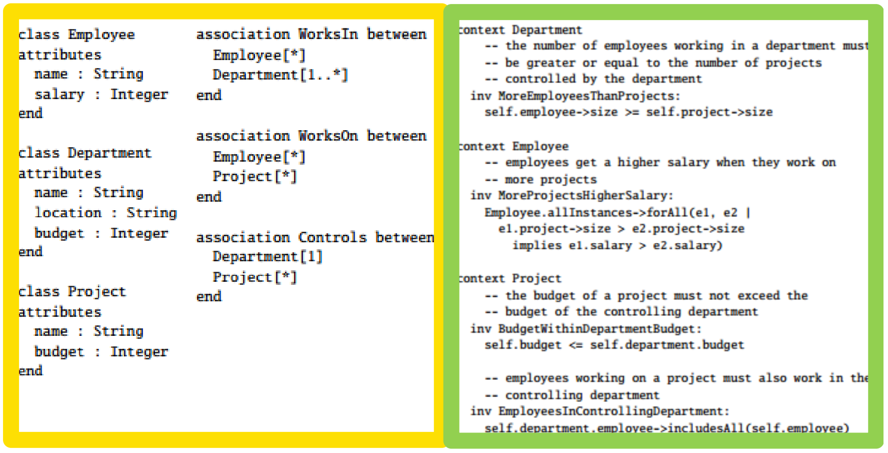
```
inv MoreEmployeesThanProjects:
  -- the number of employees working in a department must
  -- be greater or equal to the number of projects
  -- controlled by the department
  self.employee->size >= self.project->size
```

Description

```
inv BudgetWithinDepartmentBudget:
  -- The budget of a project must not exceed the
  -- budget of the controlling department.
  self.budget <= self.department.budget
```

Description

```
inv EmployeesInControllingDepartment:
  -- Employees working on a project must also
  -- work in the controlling department.
  self.department.employee
     ->includesAll(self.employee)
```



```
class Employee
attributes
  name : String
  salary : Integer
end

class Department
attributes
  name : String
  location : String
  budget : Integer
end

class Project
attributes
  name : String
  budget : Integer
end
```

```
association WorksIn between
  Employee[*]
  Department[1..*]
end

association WorksOn between
  Employee[*]
  Project[*]
end

association Controls between
  Department[1]
  Project[*]
end
```

```
ontext Department
   -- the number of employees working in a department must
   -- be greater or equal to the number of projects
   -- controlled by the department
 inv MoreEmployeesThanProjects:
   self.employee->size >= self.project->size

ontext Employee
   -- employees get a higher salary when they work on
   -- more projects
 inv MoreProjectsHigherSalary:
   Employee.allInstances->forAll(e1, e2 |
     e1.project->size > e2.project->size
       implies e1.salary > e2.salary)

ontext Project
   -- the budget of a project must not exceed the
   -- budget of the controlling department
 inv BudgetWithinDepartmentBudget:
   self.budget <= self.department.budget

   -- employees working on a project must also work in the
   -- controlling department
 inv EmployeesInControllingDepartment:
   self.department.employee->includesAll(self.employee)
```

Objectives

- Integration of OCL contraints into Modelio
  - use of a simple "Contraint" profile defined on Notes
    - <<Invariant>>,<<Precondition>>,<<Postcondition>>

- Generation of the "Structural" model
- Copy of the OCL constraints

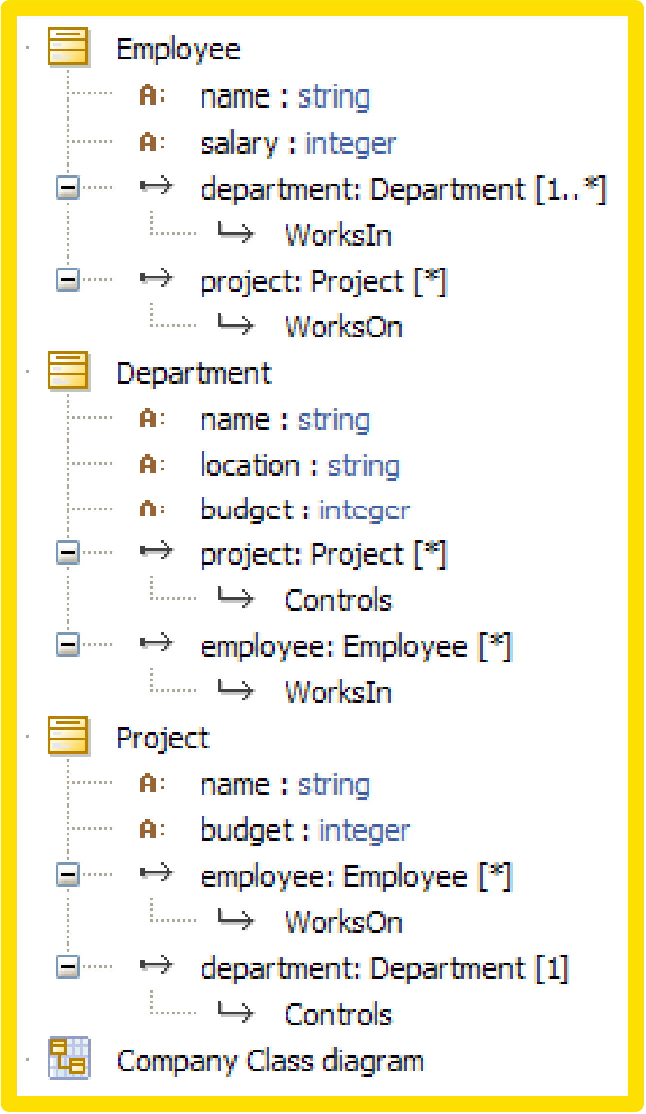
Possible Extensions

- Launching OCLUSE environment & error reporting
- Reverse engineering of OCL models
- Instances (.soil)
  - generation from object models
  - reverse engineering

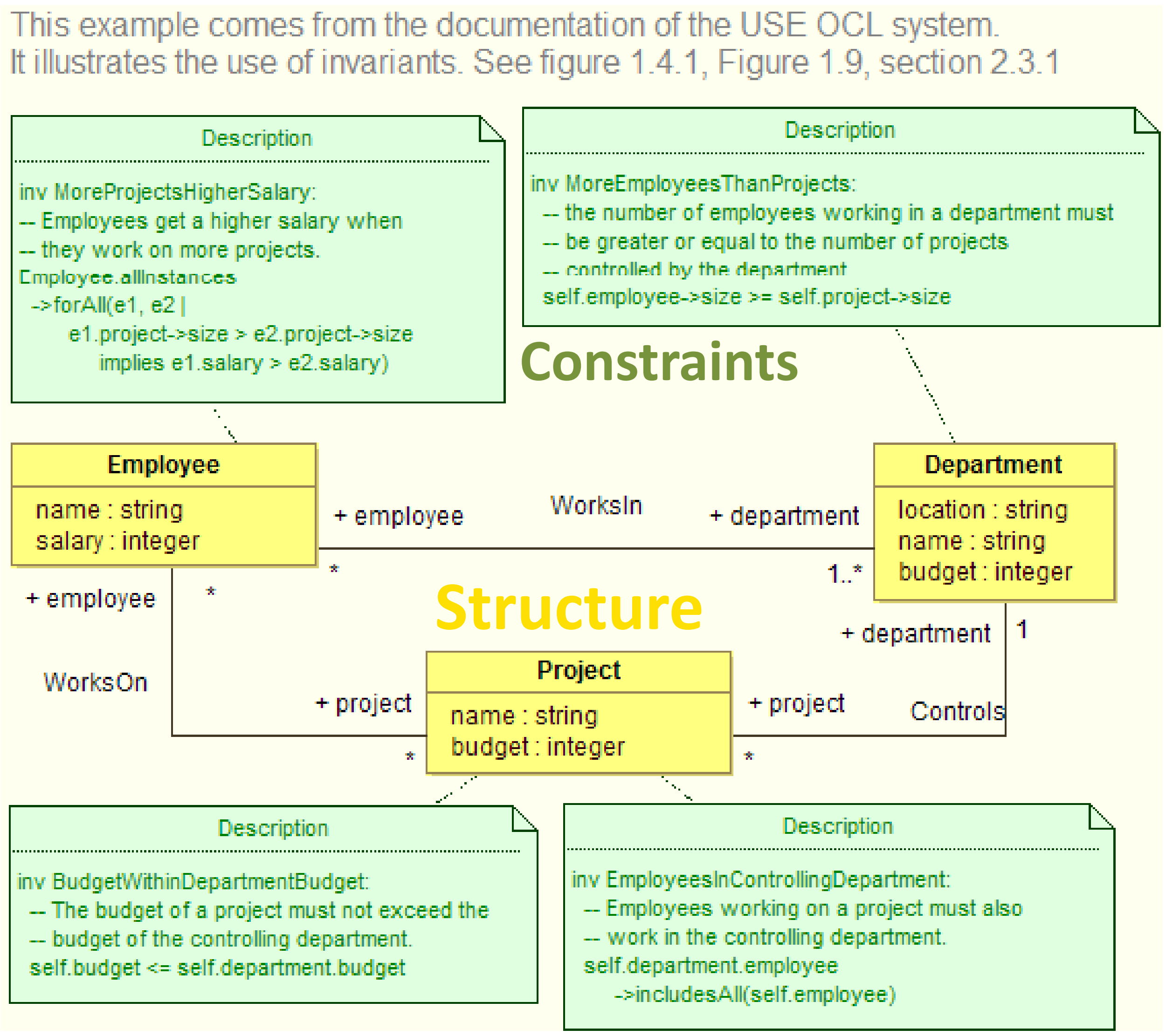
"Company" Example

MODELIO

Structure



```
Employee
    name : string
    salary : integer
    department: Department [1..*]
        WorksIn
    project: Project [*]
        WorksOn
Department
    name : string
    location : string
    budget : integer
    project: Project [*]
        Controls
    employee: Employee [*]
        WorksIn
Project
    name : string
    budget : integer
    employee: Employee [*]
        WorksOn
    department: Department [1]
        Controls
Company Class diagram
```



This example comes from the documentation of the USE OCL system.
It illustrates the use of invariants. See figure 1.4.1, Figure 1.9, section 2.3.1

Description

```
inv MoreProjectsHigherSalary:
-- Employees get a higher salary when
-- they work on more projects.
Employee.allInstances
  ->forAll(e1, e2 |
      e1.project->size > e2.project->size
         implies e1.salary > e2.salary)
```

Description

```
inv MoreEmployeesThanProjects:
  -- the number of employees working in a department must
  -- be greater or equal to the number of projects
  -- controlled by the department
  self.employee->size >= self.project->size
```

Constraints

Structure

Description

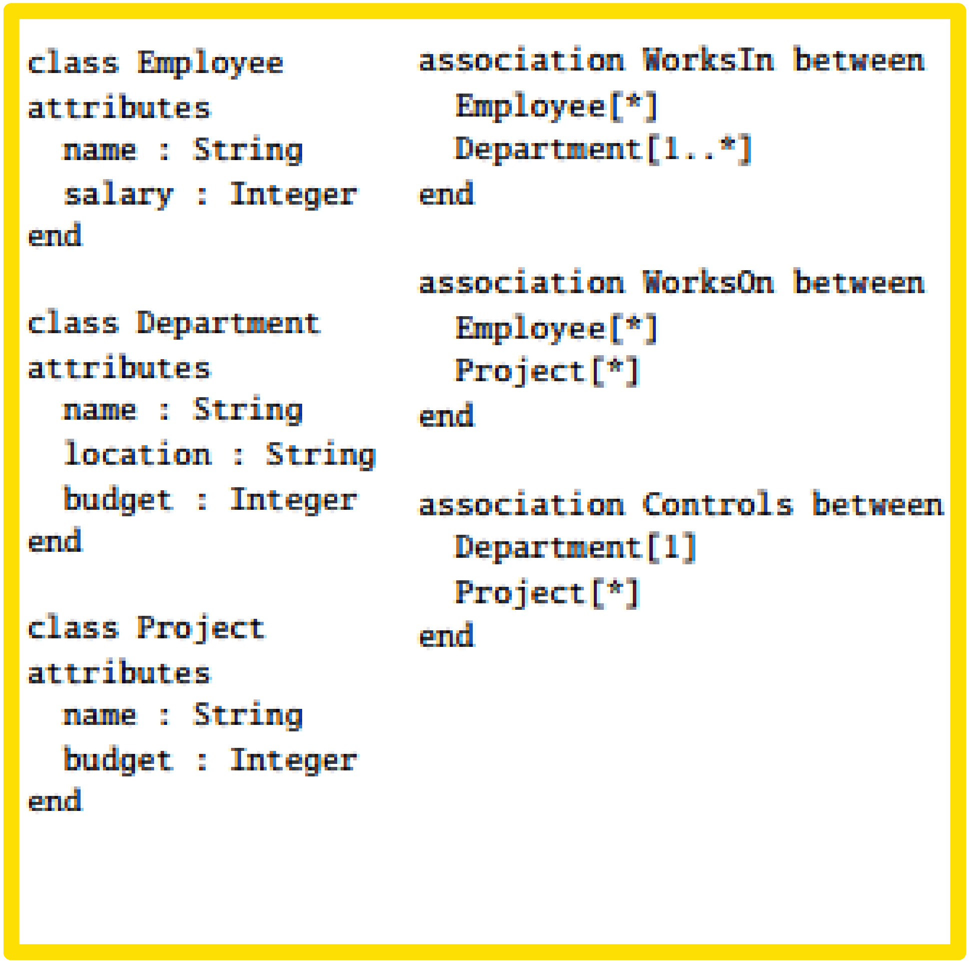
```
inv BudgetWithinDepartmentBudget:
  -- The budget of a project must not exceed the
  -- budget of the controlling department.
  self.budget <= self.department.budget
```

Description

```
inv EmployeesInControllingDepartment:
  -- Employees working on a project must also
  -- work in the controlling department.
  self.department.employee
      ->includesAll(self.employee)
```

"Company" Example

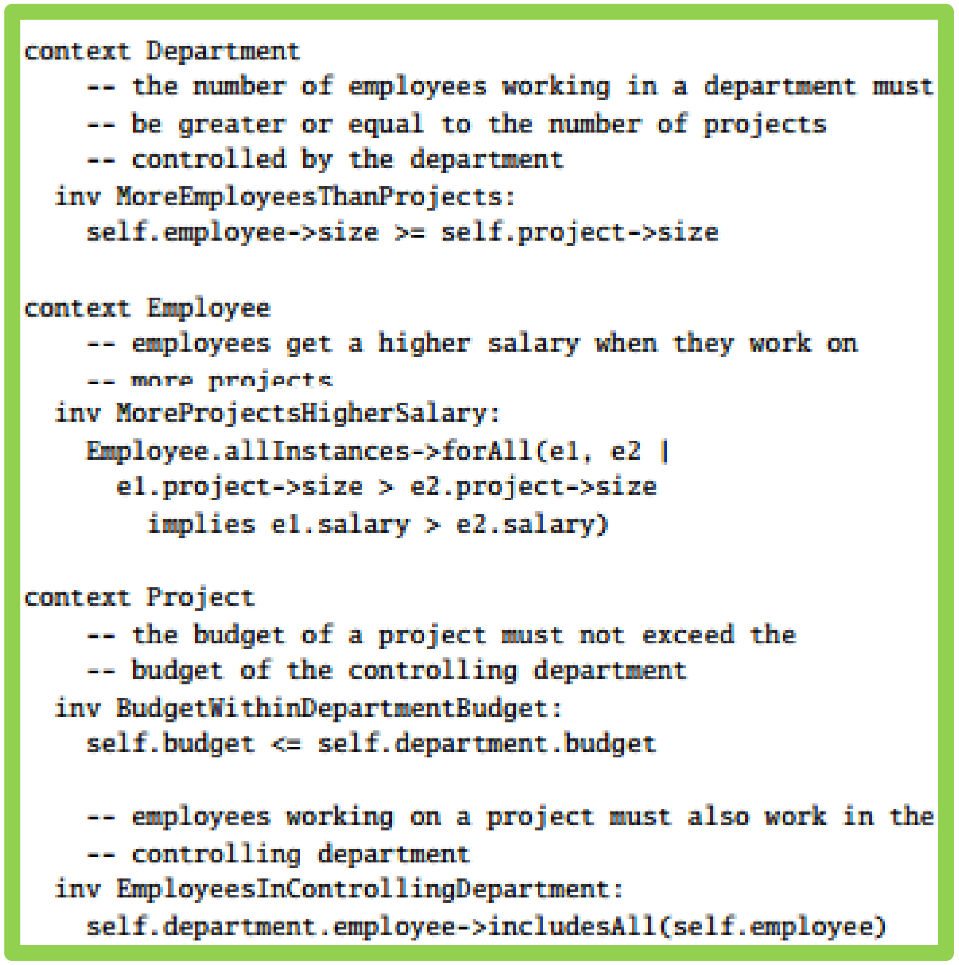
Structure



```
class Employee
attributes
  name : String
  salary : Integer
end

class Department
attributes
  name : String
  location : String
  budget : Integer
end

class Project
attributes
  name : String
  budget : Integer
end
```

```
association WorksIn between
  Employee[*]
  Department[1..*]
end

association WorksOn between
  Employee[*]
  Project[*]
end

association Controls between
  Department[1]
  Project[*]
end
```

Constraints



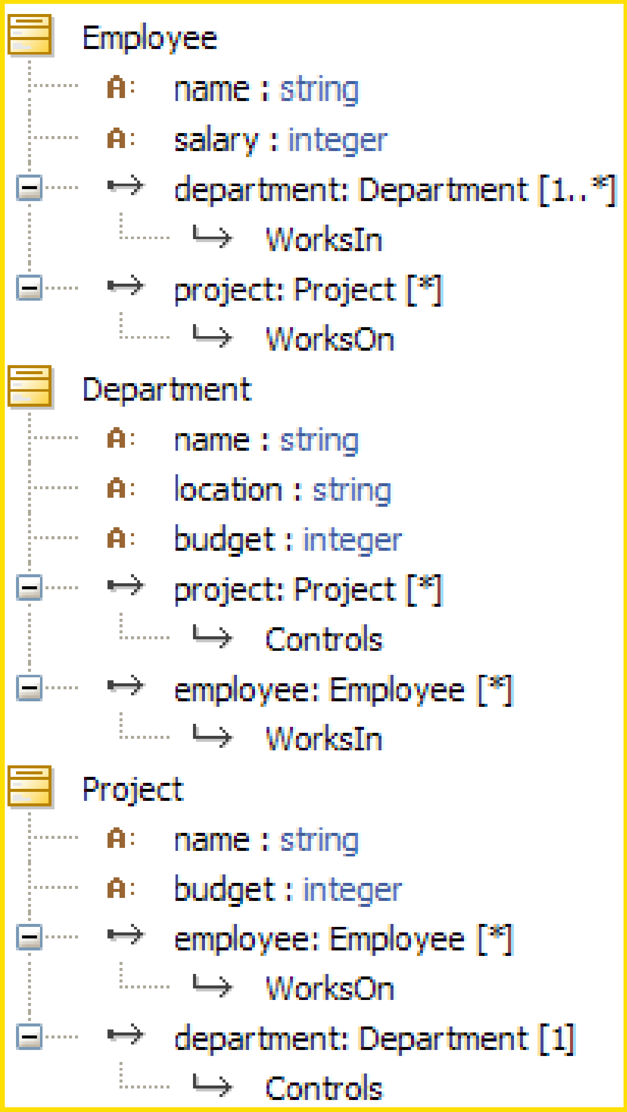
```
context Department
    -- the number of employees working in a department must
    -- be greater or equal to the number of projects
    -- controlled by the department
  inv MoreEmployeesThanProjects:
    self.employee->size >= self.project->size

context Employee
    -- employees get a higher salary when they work on
    -- more projects
  inv MoreProjectsHigherSalary:
    Employee.allInstances->forAll(e1, e2 |
      e1.project->size > e2.project->size
        implies e1.salary > e2.salary)

context Project
    -- the budget of a project must not exceed the
    -- budget of the controlling department
  inv BudgetWithinDepartmentBudget:
    self.budget <= self.department.budget

    -- employees working on a project must also work in the
    -- controlling department
  inv EmployeesInControllingDepartment:
    self.department.employee->includesAll(self.employee)
```

See section 2.3.1 of USEOCL manual for comments and reference to the full version

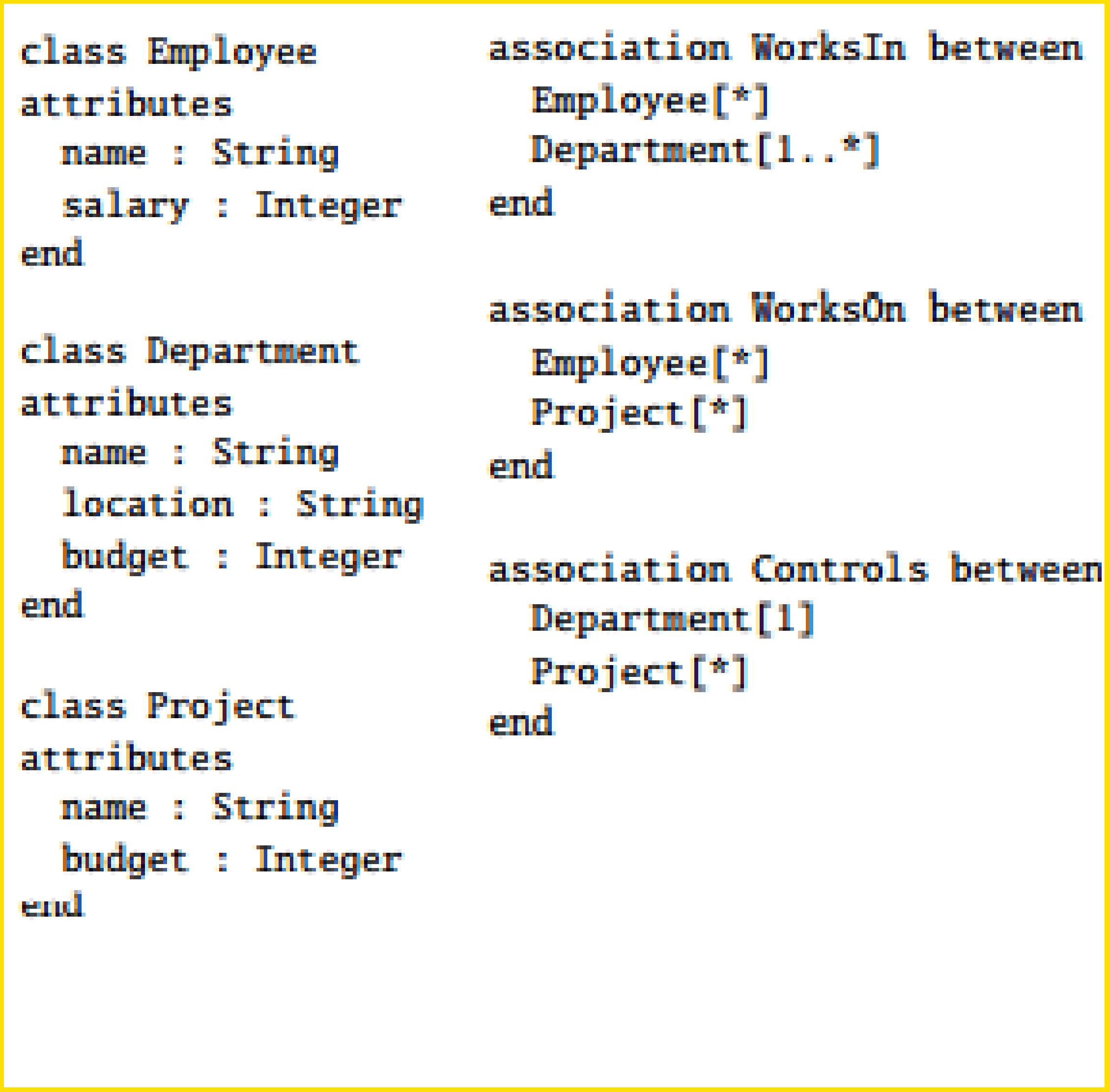
Structure

MODELIO

USE



```
Employee
    name : string
    salary : integer
    department: Department [1..*]
        WorksIn
    project: Project [*]
        WorksOn
Department
    name : string
    location : string
    budget : integer
    project: Project [*]
        Controls
    employee: Employee [*]
        WorksIn
Project
    name : string
    budget : integer
    employee: Employee [*]
        WorksOn
    department: Department [1]
        Controls
```

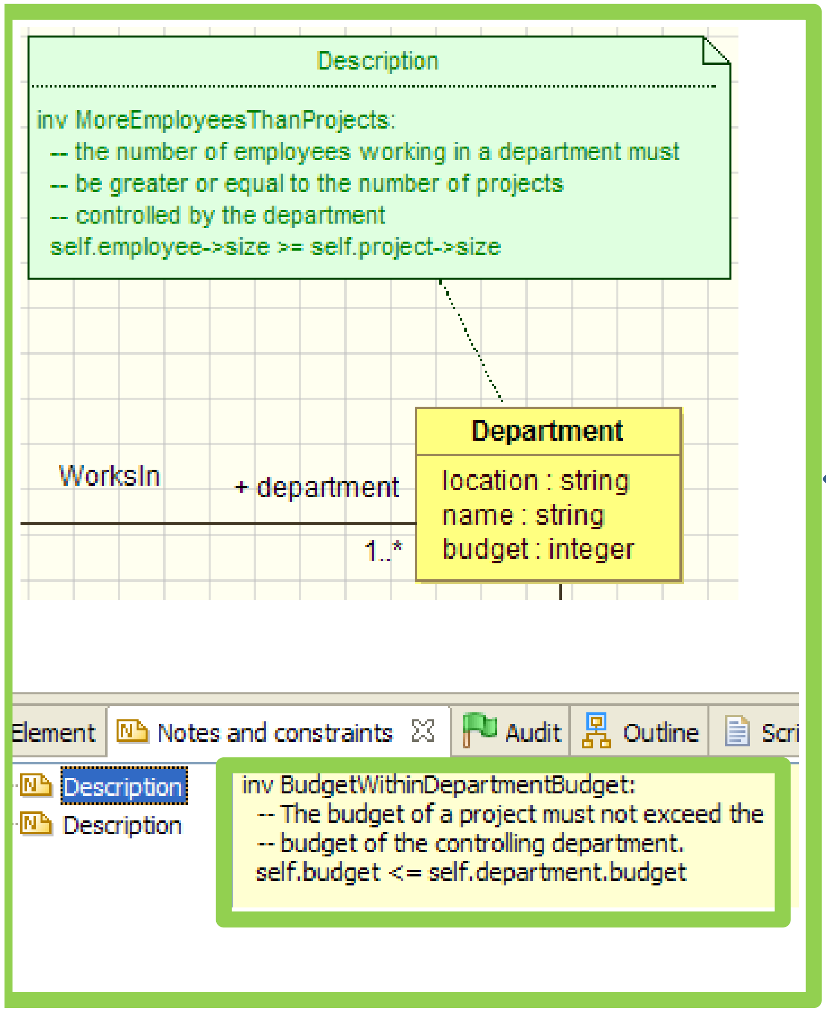


```
class Employee
attributes
  name : String
  salary : Integer
end

class Department
attributes
  name : String
  location : String
  budget : Integer
end

class Project
attributes
  name : String
  budget : Integer
end

association WorksIn between
  Employee[*]
  Department[1..*]
end

association WorksOn between
  Employee[*]
  Project[*]
end

association Controls between
  Department[1]
  Project[*]
end
```

Constraints

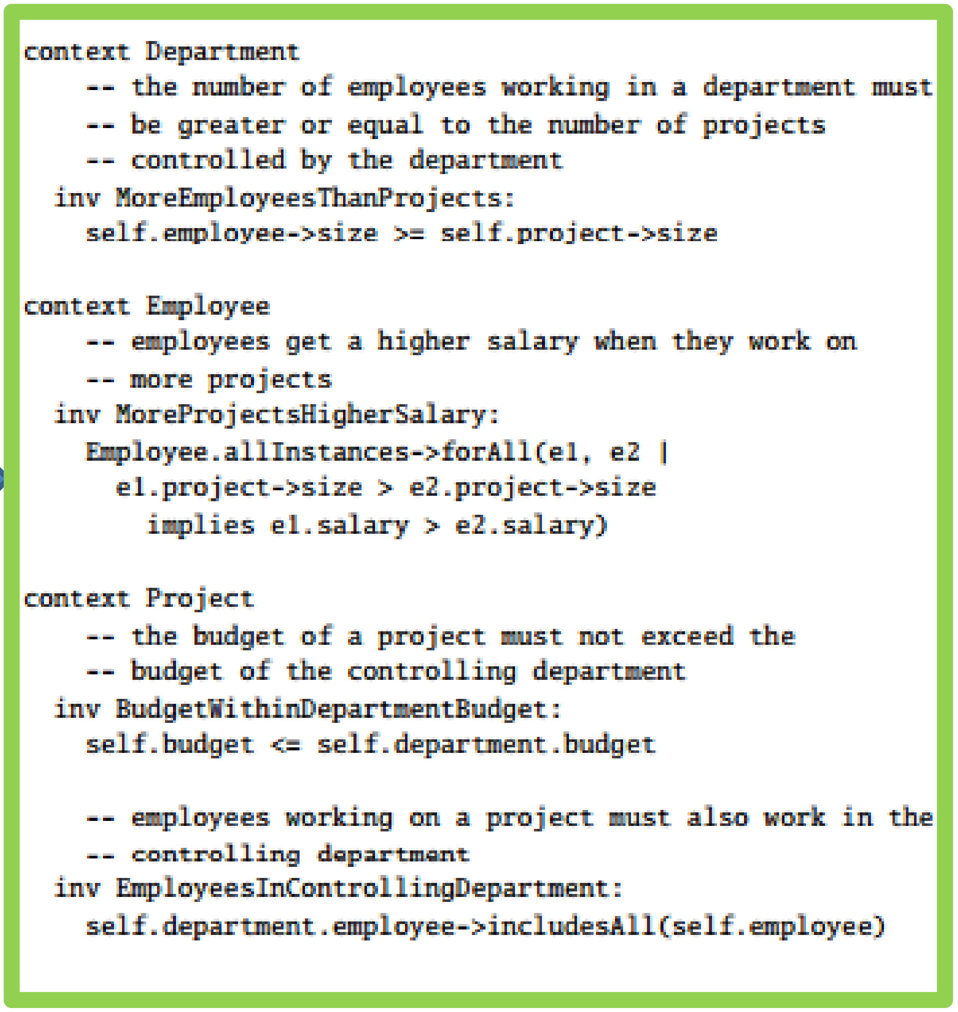
MODELIO

USE



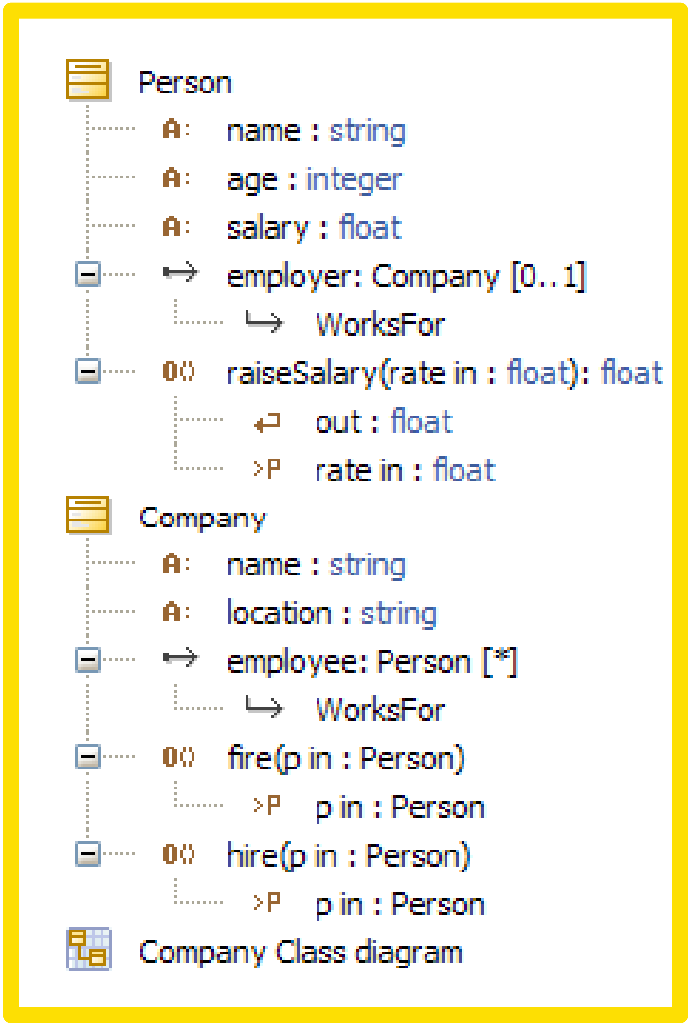
```
inv MoreEmployeesThanProjects:
  -- the number of employees working in a department must
  -- be greater or equal to the number of projects
  -- controlled by the department
  self.employee->size >= self.project->size
```

```
inv BudgetWithinDepartmentBudget:
  -- The budget of a project must not exceed the
  -- budget of the controlling department.
  self.budget <= self.department.budget
```



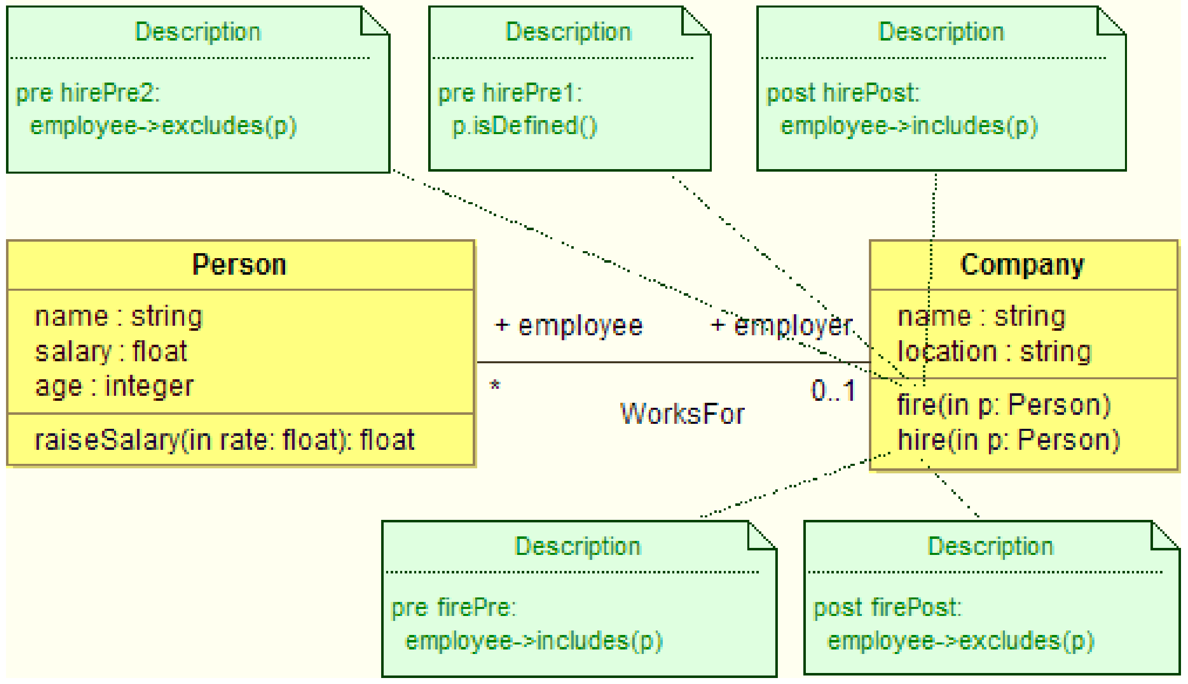
```
context Department
    -- the number of employees working in a department must
    -- be greater or equal to the number of projects
    -- controlled by the department
  inv MoreEmployeesThanProjects:
    self.employee->size >= self.project->size

context Employee
    -- employees get a higher salary when they work on
    -- more projects
  inv MoreProjectsHigherSalary:
    Employee.allInstances->forAll(e1, e2 |
      e1.project->size > e2.project->size
        implies e1.salary > e2.salary)

context Project
    -- the budget of a project must not exceed the
    -- budget of the controlling department
  inv BudgetWithinDepartmentBudget:
    self.budget <= self.department.budget

    -- employees working on a project must also work in the
    -- controlling department
  inv EmployeesInControllingDepartment:
    self.department.employee->includesAll(self.employee)
```

"Employee" Example



```
Person
    name : string
    age : integer
    salary : float
    employer: Company [0..1]
        WorksFor
    raiseSalary(rate in : float): float
        out : float
        rate in : float
Company
    name : string
    location : string
    employee: Person [*]
        WorksFor
    fire(p in : Person)
        p in : Person
    hire(p in : Person)
        p in : Person
Company Class diagram
```

This example comes from the documentation of the OCLUse system.
This example illustrates the use of operations, pre and post conditions
See figure 1.4.1, Figure 1.9, section 2.3.2



Description
pre hirePre2:
employee->excludes(p)

Description
pre hirePre1:
p.isDefined()

Description
post hirePost:
employee->includes(p)

Description
pre firePre:
employee->includes(p)

Description
post firePost:
employee->excludes(p)

MODELIO Scribes

For **downloads and contributions**

visit github.com/megaplan/ModelioScribes

github SOCIAL CODING