Usage of OCL

for design guidelines of models

Christian Hein, Fraunhofer FOKUS

April 2006

Context of this work

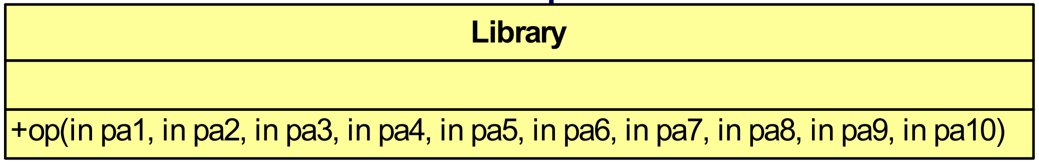
- The present courseware has been elaborated in the context of the MODELWARE European IST FP6 project (http://www.modelware-ist.org/).
- Co-funded by the European Commission, the MODELWARE project involves 19 partners from 8 European countries. MODELWARE aims to improve software productivity by capitalizing on techniques known as Model-Driven Development (MDD).
- To achieve the goal of large-scale adoption of these MDD techniques, MODELWARE promotes the idea of a collaborative development of courseware dedicated to this domain.
- The MDD courseware provided here with the status of open source software is produced under the EPL 1.0 license.

Overview

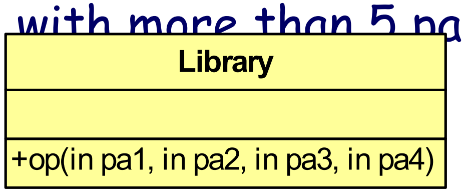
- design guidelines for models
- example and practical demonstration

design guidelines for models

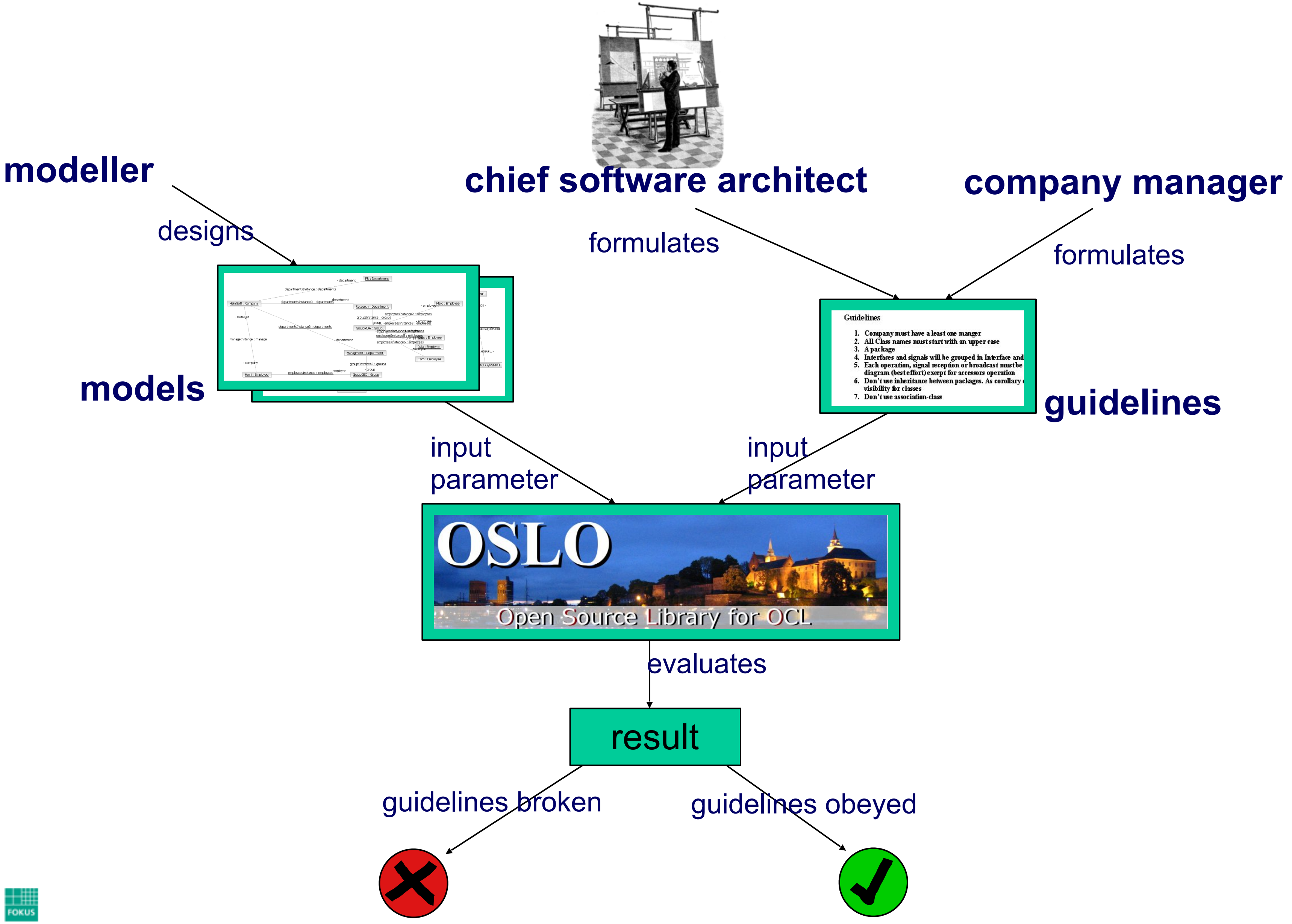
- guidelines can be expressed as OCL constraints
- belong to a model in an indirect way
- must not be strict, can be broken without consequence
- can be changed over time (different from model to model)
- clearness of models
  - models are more readable and plain


Library
+op(in pa1, in pa2, in pa3, in pa4, in pa5, in pa6, in pa7, in pa8, in pa9, in pa10)

- improve quality of models
  - not allow operations with more than 5 parameters



design guidelines for models



modeller

chief software architect

company manager

designs

formulates

formulates

models

guidelines

Guidelines

1. Company must have a least one manger
2. All Class names must start with an upper case
3. A package
4. Interfaces and signals will be grouped in Interface and
5. Each operation, signal reception or broadcast must be diagram (best effort) except for accessors operation
6. Don't use inheritance between packages. As corollary visibility for classes
7. Don't use association-class

input parameter

input parameter

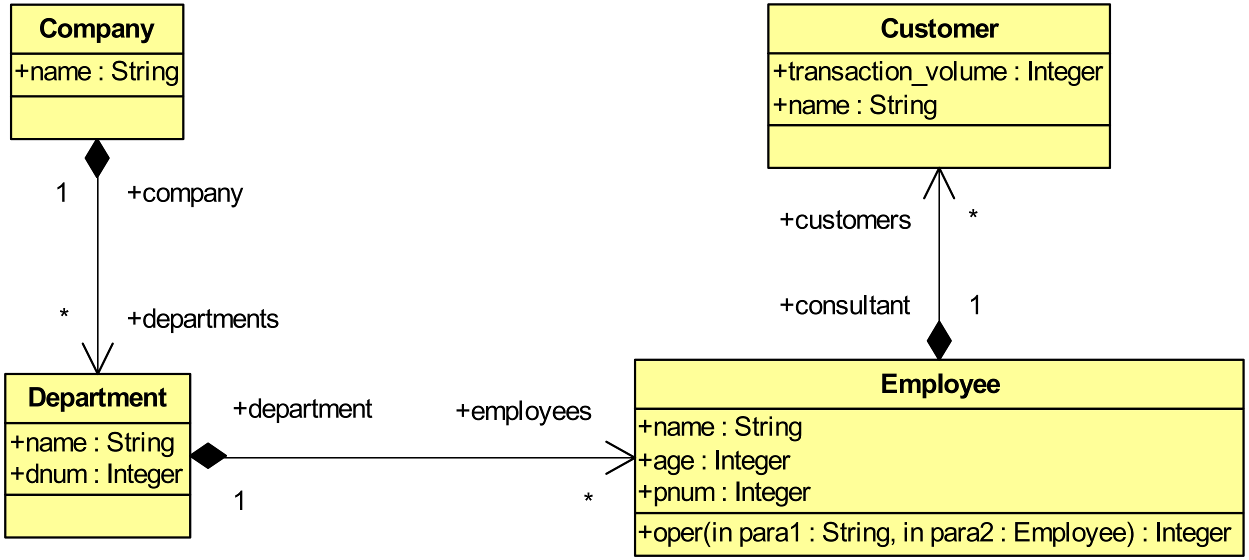
evaluates

result

guidelines broken

guidelines obeyed

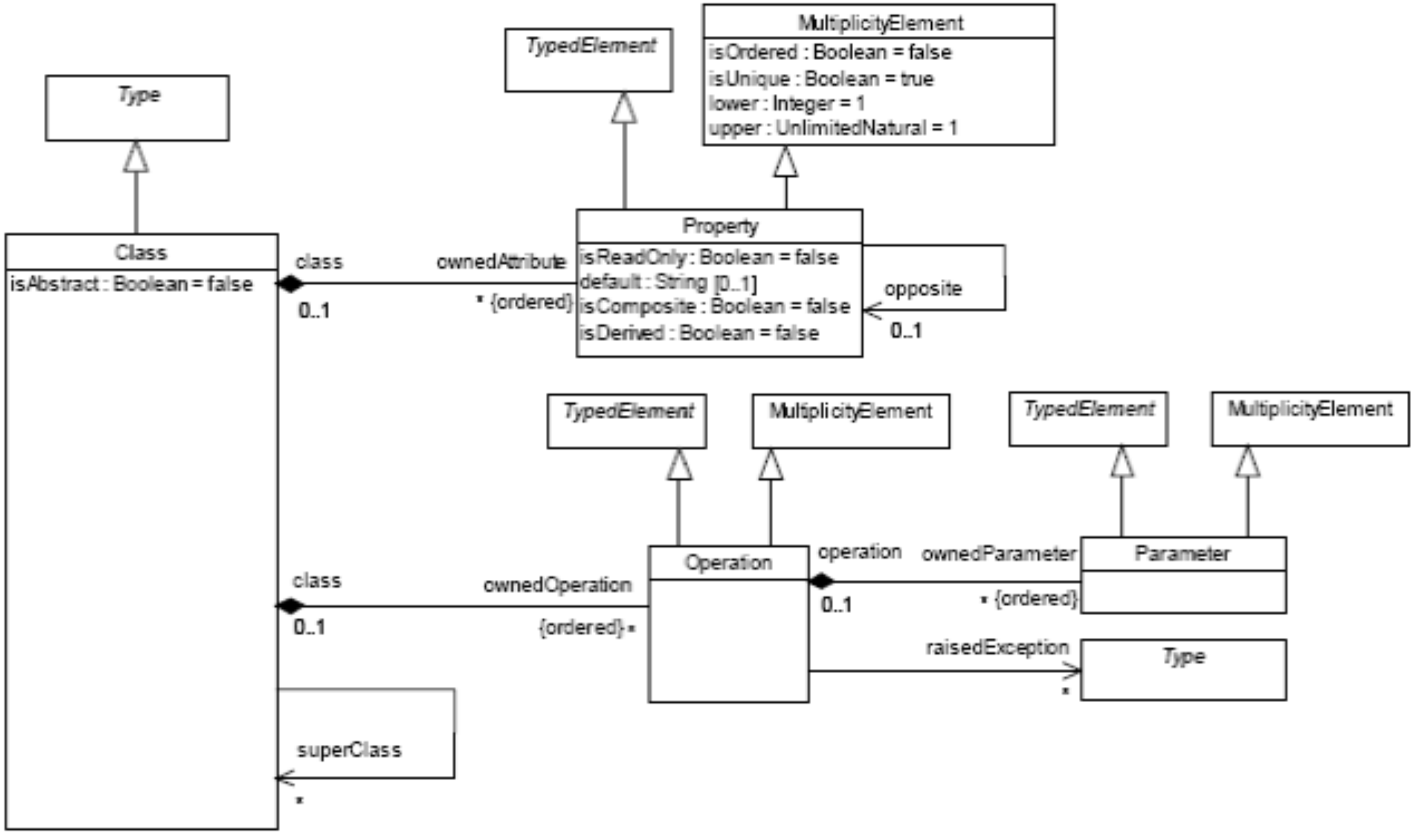
example and practical demonstration



- company model based on UML2
- design guidelines from the chief software architect
  1. all class names have to start with an upper case
  2. an operation do not have more than 2 parameters

example and practical demonstration

- translate the constraint from natural language into OCL, therefore we need a snapshot of the UML2 metamodel



example and practical demonstration

- all class names have to start with an upper case

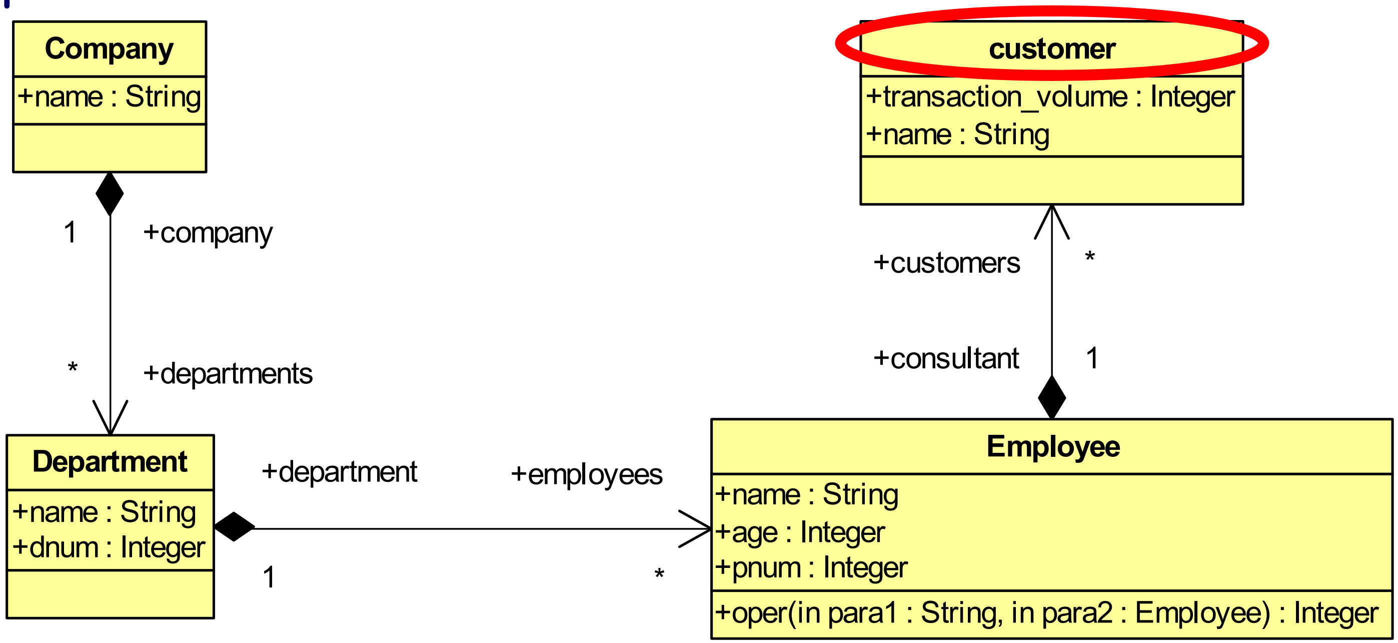
```
context Class inv:
let upper:Set(String)=Set{'A','B','C','D','E','F',
'G','H','I','J','K','L','M','N','O','P','Q','R',
'S','T','U','V','W','X','Y','Z'} in
upper->includes(name.substring(1,1))
```

- an operation do not have more than 2 parameters

```
context Operation inv: ownedParameter->size()<3
```

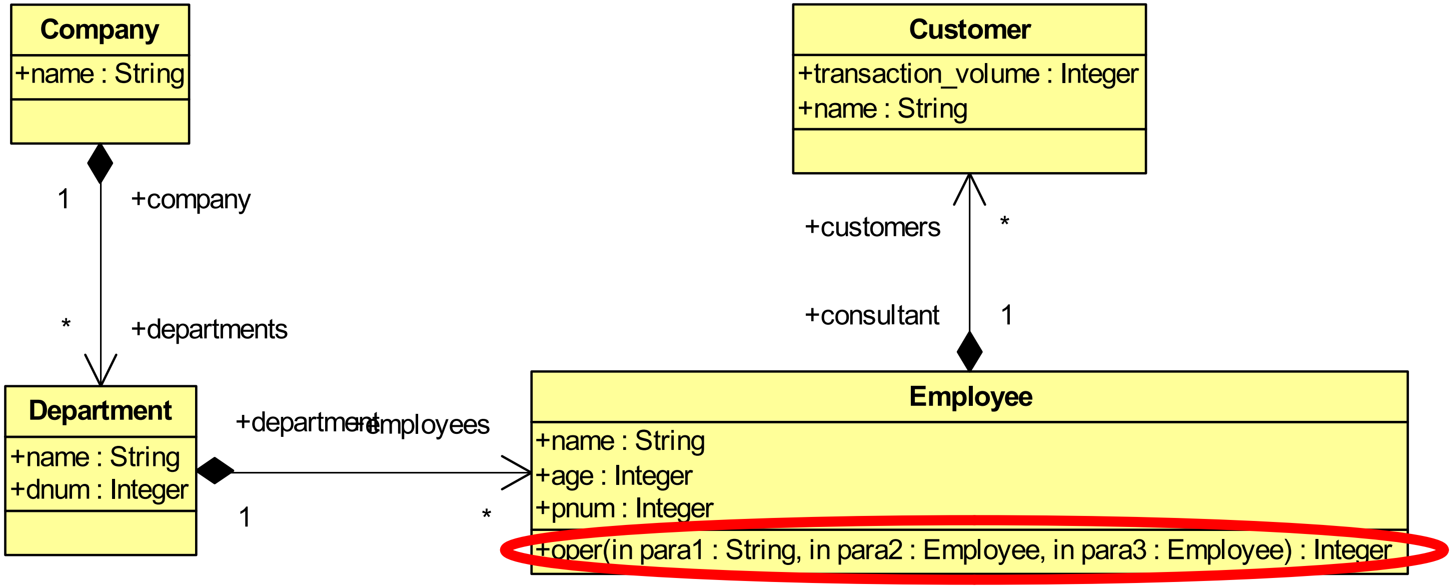
example and practical demonstration

- three cases are demonstrated
- first case: both guidelines are obeyed
- second case: it exists a class which doesn't start with an upper case:



example and practical demonstration

- third case: it exists an operation which owns more than 2 parameters



Summary

- OSLO based on the Kent OCL library
  - http://oslo-project.berlios.de
- the shown functionality is also reachable over the modelbus
- metrics can be handled similar to design guidelines