should future variability modeling languages express constraints in ocl?

Don Batory
Department of Computer Science
University of Texas at Austin
September 2019

No!

The End

not altogether my idea, unfortunately…

- I thank a referee of my paper for the humor of my introduction

> Beyond the question that serves as title (spoiler: the answer is no),

seriously, no.

- MDE tooling has long been a problem
- Ideas behind OCL (Object Constraint Language) are good
  - design dreadful – nothing simple or elegant about it
  - OMG definition is 246 pages…
  - OCL is pfft…

- So what?
  - This workshop is on the future of variability languages
  - And next generation Feature Modeling Languages
  - Do I know what the future holds in this area?

No!

next generation feature models

- I do know that NGFMs will support:
    - Features with attributes
    - Numerical features
    - Numerical constraints
    - Feature replication….
- I recoiled at recent attempts
    - CVL (Common Variability Language) – used OCL for constraints…
- Don't know the answer, I do know answer should be guided by 3 principles…

principle #1: simplicity!

- **Propositional Logic** was chosen for classical FM constraints because it was a simple mathematical standard

- Not sure there is a formal language for it;
  bottom line: hard to screw-up writing prop-logic constraints

- Ideally NGFM constraints should be simple to write

principle #2: don't invent, reuse!

- Do we really need a new constraint language for next generation FMs?
- Clearly we need more than prop-logic
- But are we good enough as **language engineers** to create a new constraint language without making a complete mess of it?
    - authors of OCL were not experts in language engineering…
- Our expertise is in product line engineering, **not language engineering!**

No!

principle #3: circularity avoidance

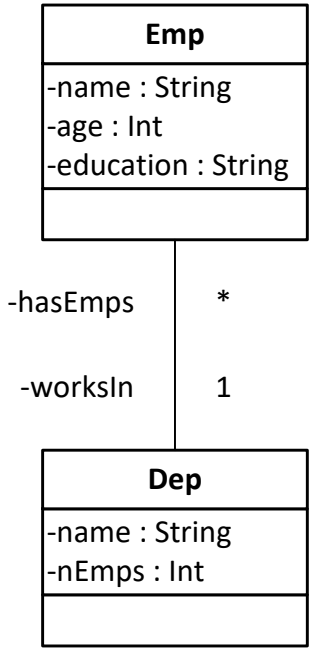
- As soon as we generalize beyond hierarchical relationships of classical FMs means we are not far away from UML class diagrams and MDE meta-models (= class diagram + constraints)

- Begs the use of OCL, and we're back to square one…

- Let me show you some recent work that could get us out of this mess
  - **It may not be needed for next generation FMs**
  - Or it might… I don't know

aocl

- Aocl is based on allegories, a branch of category theory with powerset domains
- That's its origin. Took me about 4-years to put it all together
- The core of OCL is relational algebra written in OO syntax with customized names for right-semijoins. This language is Aocl; implemented in pure Java.



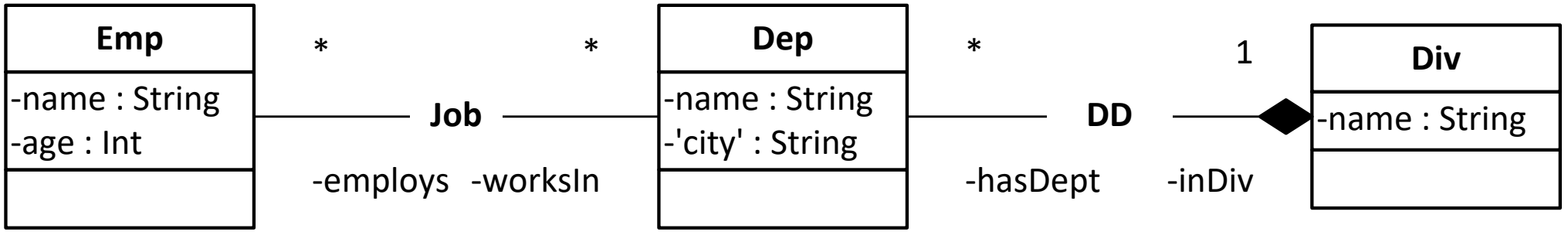
Print employees named 'don'

```
Emp.select(t->t.name.equals("don")).print();
```

Print employees that work in book department(s)

```
Dep.select(t->t.name.equals("book")).hasEmps().print()
```

more queries

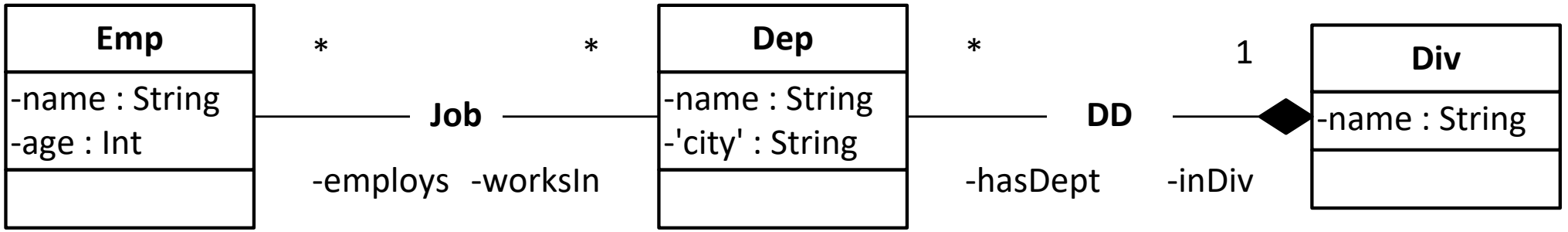


Find divisions in which 'don' works

```
Emp.select(e->e.name.equals("don")).worksIn().inDiv().print();
```

```
Emp.allInstances->select(name=`don').worksIn.inDiv
```

more queries

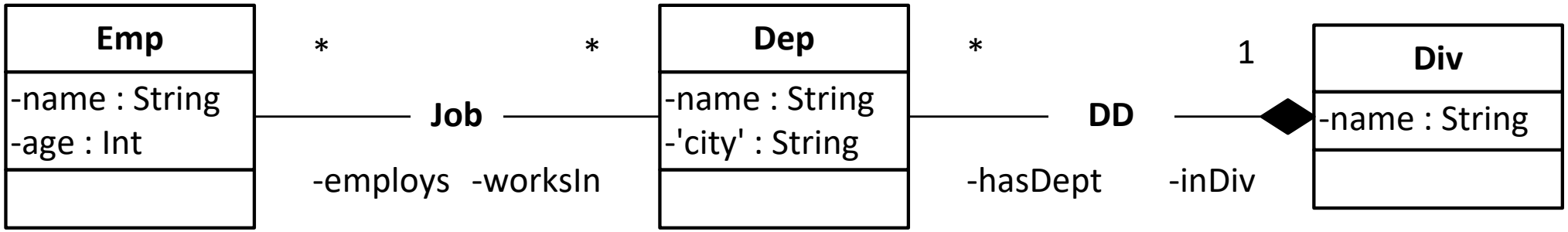


Find division colleagues of 'priscila'

```
Emp.select(e->e.name.equals("priscila"))
    .worksIn().inDiv().hasDept().employs().print();
```

```
Emp.allInstances->select(name='priscila')
                .worksIn.inDiv.hasDept.employs
```

constraints



Every Dep in Toronto must hire workers 19 and older

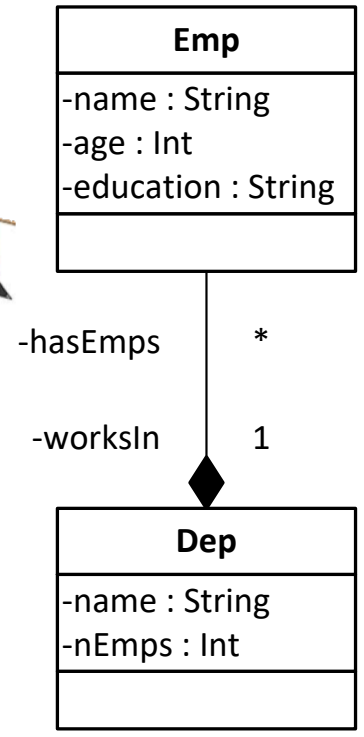
```
Dep.select(d->d.city.equals("toronto"))
    .forEach(d->d.employs().select(e->e.age<19)
    .error("%s illegally hired %s", e->d.name, e->e.name));
```

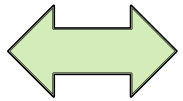
```
context Dept inv EmpAge:
self.select(city=`Toronto')
    .employs->forAll(e|e.age>=19)
```

aocl provides better error reporting than just yes/no

how does aocl work?

- Aocl is a plug-in to a Java framework
- You draw or write your class diagram and tools generate the plugin
- You then write your queries and constraints

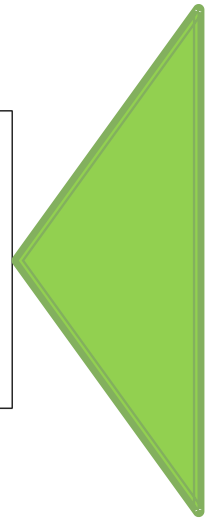




```
classDiagram ed.

table(Emp,[name,age:int,education]).
table(Dep,[name,nEmps:int]).

Assoc : Dep worksin BLACK_DIAMOND
                -> Emp hasEmps BLANK.
```

from this mde tools produce

```
classDiagram ed.

table(Emp,[name,age:int,education]).
table(Dep,[name,nEmps:int]).

Assoc : Dep worksin BLACK_DIAMOND,
        -> Emp hasEmps BLANK.
```



- Emp class (instances are tuples)
- EmpTable class (instances are tables of Emps)
- Dep class (instances are tuples)
- DepTable class (instances are tables of Deps)
- Database class (instances are ed databases)
- Database schema**
  for main-memory MDE tools called MDELite

benefits

Pure Java

Use Pure Java IDE

No special parser

Lower overhead to adopt

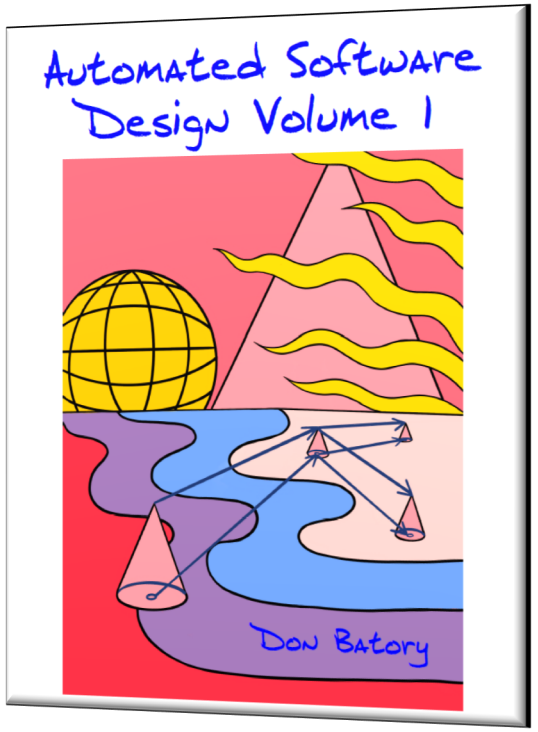
Lower overhead to learn

Lower overhead to maintain

Announcements!

aocl almost available now at my web page!

- Am seeking partners to run an empirical experiment on using OCL vs Aocl
- Interest, anyone? See me afterwards…



Discusses AOCL and much more

Free to Students & Faculty

The End