

Improvements to OCL Implementation within the Monti-Core workbench

Ferdinand Mehlan

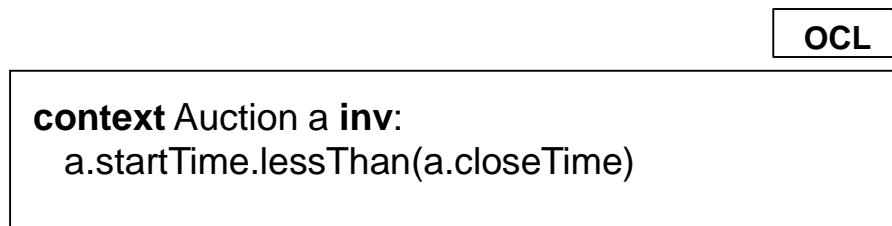
Seminar

am Lehrstuhl für Software Engineering

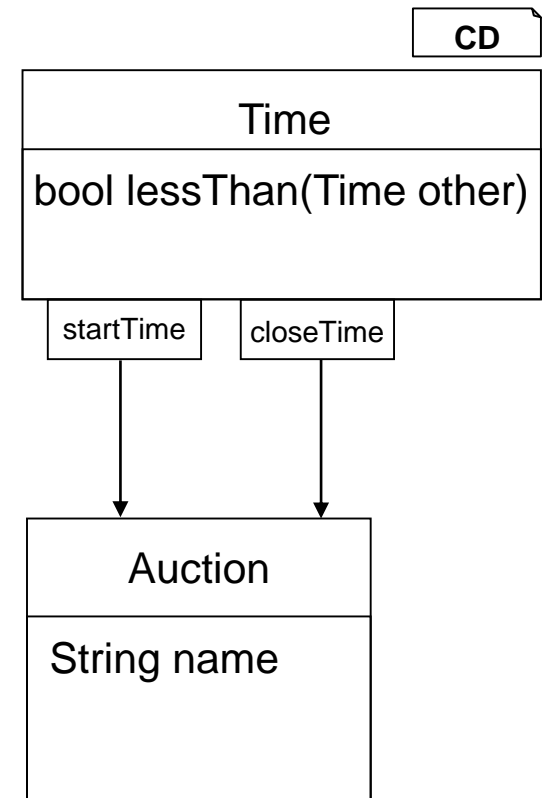
RWTH Aachen

Introduction to OCL

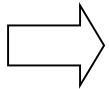
- Object Constraint Language (**OCL**)
- Part of UML
- Define **invariants** on runtime objects
- Uses information of classdiagrams



- Formulated on **class-level**,
semantics applied on **object-level**



Outline



1.

Improvements: Grammar

2.

Improvements: Types in OCL

3.

Online Tool

Improvements: Grammar

- **MontiCore** language workbench
- For Domain Specific Languages (**DSLs**)
- Generate from Grammar (Parser, AST, Symboltable, Tools)

grammar

```
grammar OCL {  
  CompilationUnit =  
    ("package" package:(Name || ".")+ ";")?  
    (ImportStatement)*  
    OCLFile;  
  // ...  
}
```

OCL

```
package example.ocl;  
import example.cd.Auction;  
  
ocl myOCLDefs{  
  // ...  
}
```

- Minor Improvements:
 - Don't parse empty strings
 - Fix disparities to specification

Improvements: Grammar

grammar

- Expressions
 - Rules that can be nested
 - E.g. $\text{expr} + \text{expr}$, $!\text{expr}$

- Monticore 3.x

- No left recursion
- Nodes dependent on each other
- Changes break tools
- Strict hierarchy
- Specific Nodes for different expressions

```
grammar OCL {
```

```
  PlusMinusExpr implements Expression =
```

```
    left:PlusMinusExpr operator:["+" | "-"] right:PlusMinusExpr  
    |  
    "(" paren: PlusMinusExpr ")" | nextHigherPrec: MultDivModExpr  
    ;
```

```
  MultDivModExpr implements Expression =
```

```
    left:MultDivModExpr operator:["*" | "/" | "%"] right:MultDivModExpr  
    |  
    "(" paren: MultDivModExpr ")" | nextHigherPrec:LogicalORExpr  
    ;  
}
```

Improvements: Grammar

- Monticore 4.x
 - Direct left recursion
 - Cleaner grammar
 - All expressions in one rule
 - Not extensible
 - No visitor pattern on different expressions for tools

grammar

```
grammar OCL {  
    Expression =  
        Expression operator:["*" | "/" | "%"] Expression  
        |  
        Expression operator:["+" | "-"] Expression;  
}
```

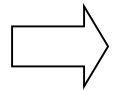
Improvements: Grammar

- Monticore 4.5.3+
 - Indirect left recursion
 - Easy to read
 - Priorities for expressions
 - Visitor pattern on each node for tools
 - Extensible

grammar

```
grammar OCL {  
  interface Expression;  
  
  PlusExpression implements Expression<20> =  
    Expression "+" Expression;  
  
  MinusExpression implements Expression<20> =  
    Expression "-" Expression;  
  
  MultExpression implements Expression<30> =  
    Expression "*" Expression;  
  
  DivExpression implements Expression<30> =  
    Expression "/" Expression;  
}
```

Improvements: Types in OCL



1.

Improvements: Grammar

2.

Improvements: Types in OCL

3.

Online Tool

Types in OCL

- OCL a type-less language
- Uses CD information to navigate
- **Concept**: Automatically check correct navigation

OCL

context Auction a **inv**:

let

b = a.bidder; // Set<Person>

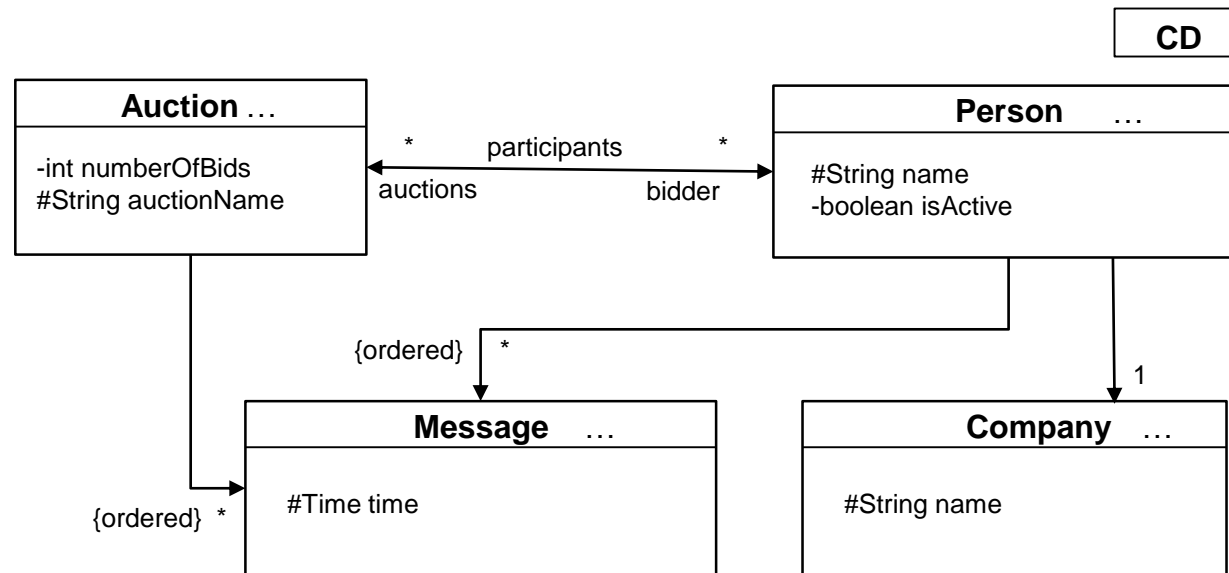
m = b.message; // List<Message>

t = {mess.time | mess in m}; // Collection<Time>

in

b == this.message

Types in OCL



OCL

```

context Auction inv:
  let
    b = a.bidder; // Set<Person>
    m = b.message; // List<Message>
    t = {mess.time | mess in m}; // Collection<Time>
  in
    b == this.message
  
```

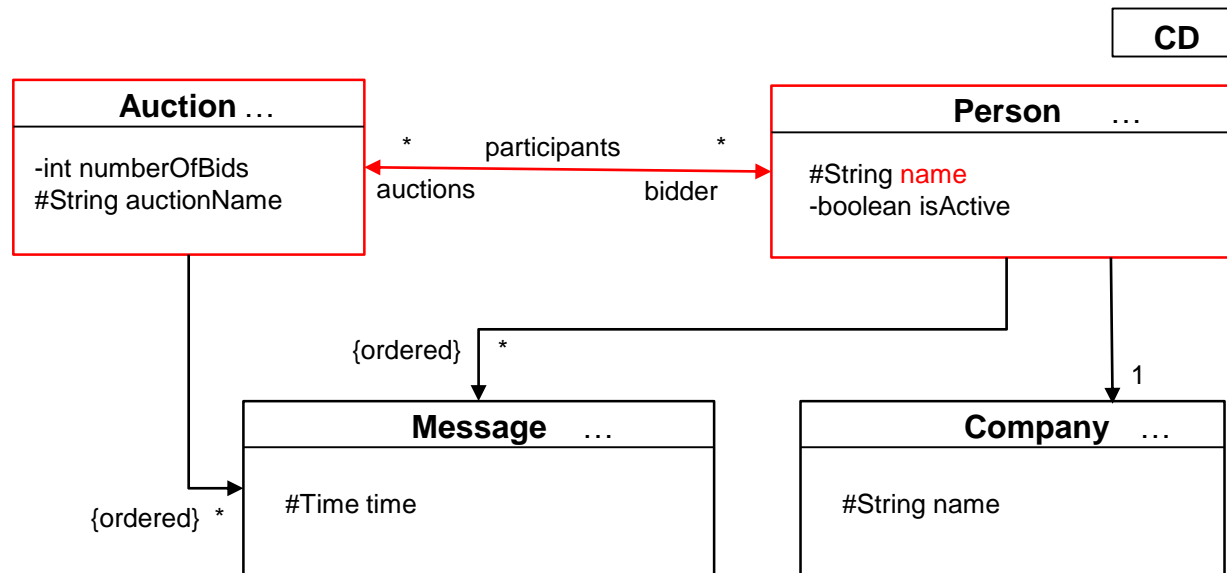
Types in OCL

- **Navigation** with the point operator using:
 - Association names, role names, class names, class fields
 - Specification mode (arrow direction is ignored)
 - Implicit variables (this, super)
 - Implicit flattening

OCL

```
context Auction a inv:  
    a.participants.name != Set{}  
context Auction a inv:  
    a.bidder.name != Set{}  
context Auction a inv:  
    a.person.name != Set{}
```

Types in OCL



OCL

```

context Auction inv:
  a.participants.name != Set{}
context Auction inv:
  a.bidder.name != Set{}
context Auction inv:
  a.person.name != Set{}
  
```

Types in OCL

- Build **OCL symboltable** and infer types
- Symbol **OCLVariable** for variable name and type
- Symbol **CDType** for type
- Automatically resolve types from **CD symboltable**
 - Check for illegal navigation

```
1 package ocl;  
2 import cd.AuctionCD;  
3 ocl myRule {  
4  
5     context Auction a inv:  
6  
7     let  
8         m1 = a.message;  
9         m2 = List{ m | m in a.bidder.message } ;  
10  
11     in  
12         m1 == m2  
13 }
```

OCLVariable (points to `a`)

CDType (points to `Auction`)

Types in OCL

- Inferring type of m2:
 - Infer type of m
 - Infer type of a.bidder.message
- M2 is of type List<Message>

```
1 package ocl;  
2 import cd.AuctionCD;  
3 ocl myRule {  
4  
5     context Auction a inv:  
6  
7     let  
8         m1 = a.message;  
9         m2 = List{ m | m in a.bidder.message } ;  
10  
11     in  
12         m1 == m2  
13 }
```

OCLVariable (points to `a`)

CDType (points to `Auction`)

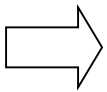
Online Tool

1.

Improvements: Grammar

2.

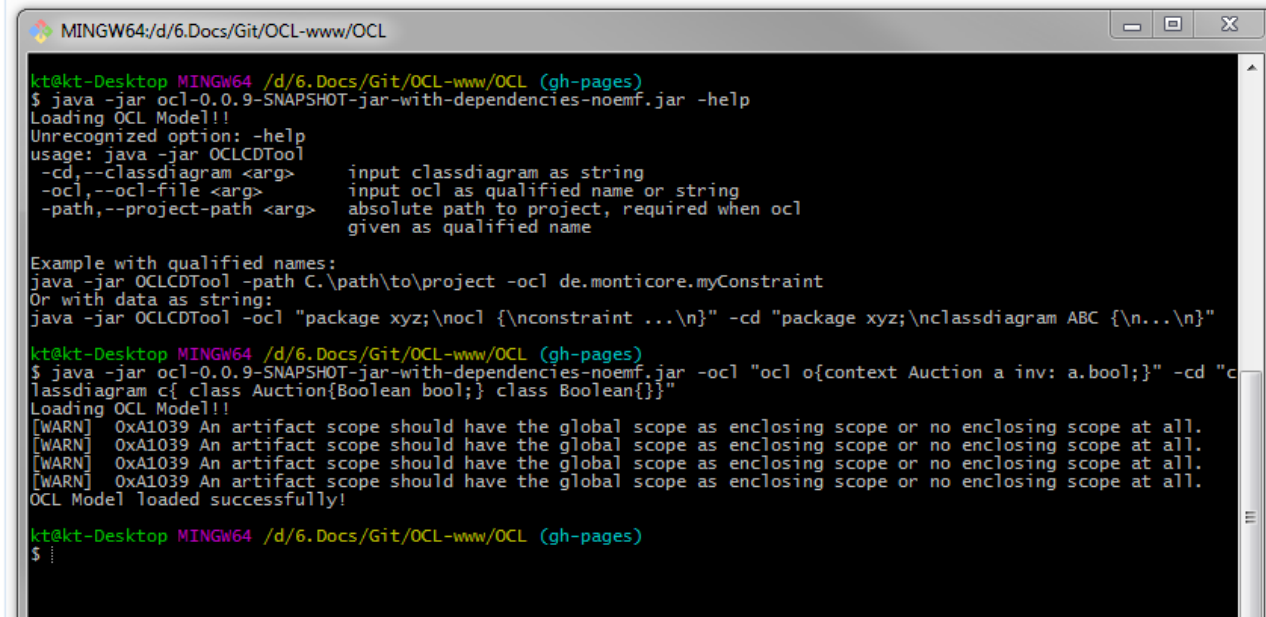
Improvements: Types in OCL



3.

Online Tool

Online Tool



```
MINGW64/d/6.Docs/Git/OCL-www/OCL

kt@kt-Desktop MINGW64 /d/6.Docs/Git/OCL-www/OCL (gh-pages)
$ java -jar ocl-0.0.9-SNAPSHOT-jar-with-dependencies-noemf.jar -help
Loading OCL Model!!
Unrecognized option: -help
usage: java -jar OCLCDTool
       -cd,--classdiagram <arg>      input classdiagram as string
       -ocl,--ocl-file <arg>         input ocl as qualified name or string
       -path,--project-path <arg>    absolute path to project, required when ocl
                                     given as qualified name

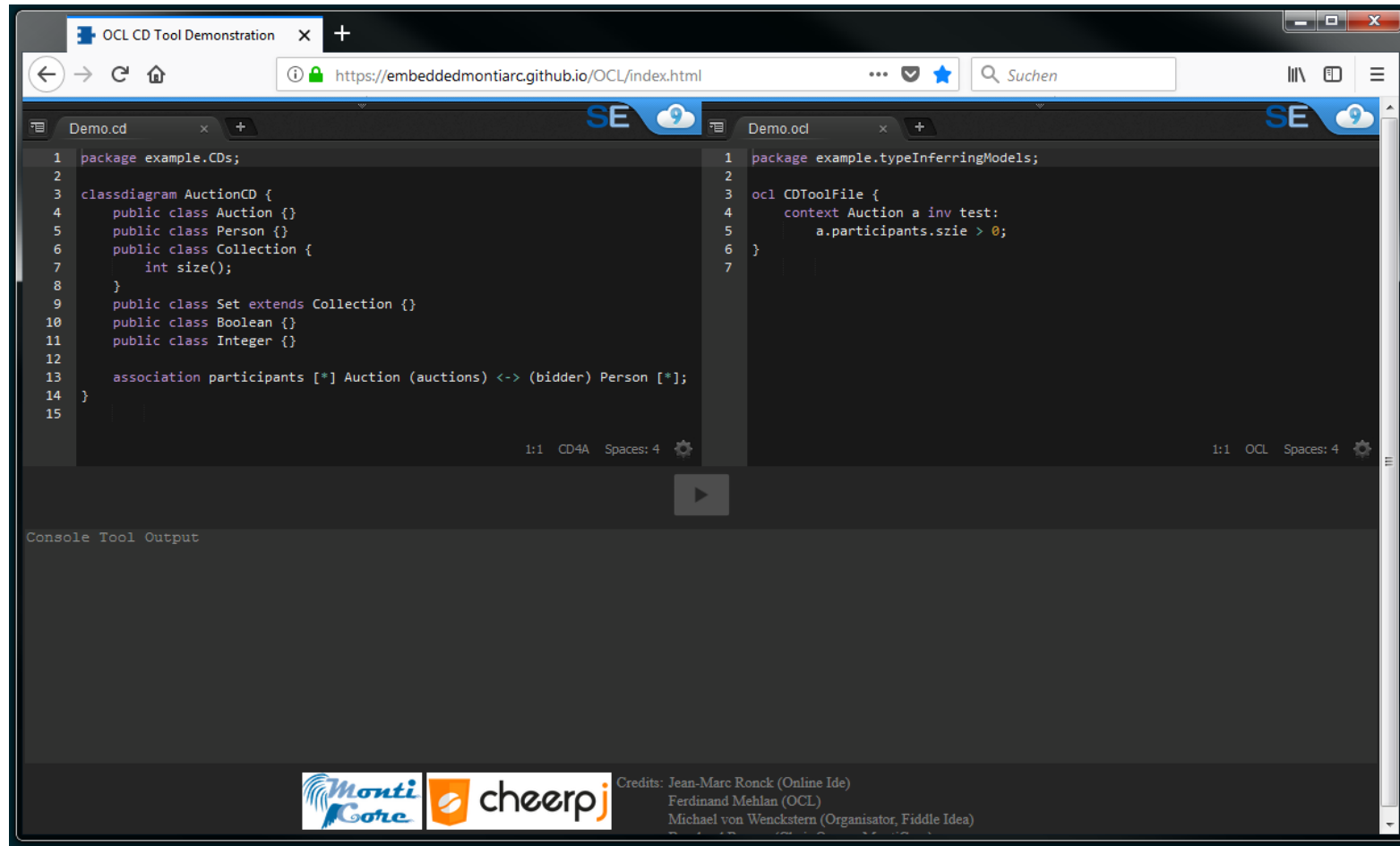
Example with qualified names:
java -jar OCLCDTool -path C:\path\to\project -ocl de.monticore.myConstraint
Or with data as string:
java -jar OCLCDTool -ocl "package xyz;\nocl {\nconstraint ... \n}" -cd "package xyz;\ncl
lassdiagram ABC {\n... \n}"

kt@kt-Desktop MINGW64 /d/6.Docs/Git/OCL-www/OCL (gh-pages)
$ java -jar ocl-0.0.9-SNAPSHOT-jar-with-dependencies-noemf.jar -ocl "ocl o{context Auction a inv: a.bool;}" -cd "c
lassdiagram c{ class Auction{Boolean bool;} class Boolean{}}"
Loading OCL Model!!
[WARN] 0xA1039 An artifact scope should have the global scope as enclosing scope or no enclosing scope at all.
[WARN] 0xA1039 An artifact scope should have the global scope as enclosing scope or no enclosing scope at all.
[WARN] 0xA1039 An artifact scope should have the global scope as enclosing scope or no enclosing scope at all.
[WARN] 0xA1039 An artifact scope should have the global scope as enclosing scope or no enclosing scope at all.
OCL Model loaded successfully!

kt@kt-Desktop MINGW64 /d/6.Docs/Git/OCL-www/OCL (gh-pages)
$
```

- **CLI Tool** to load and verify correct OCL
- Idea: Integrate tool into online interface
- **CheerpJ** by learningTech: Java -> JS transpiler
- **Online IDE** by Jean-Marc Ronck

Online Tool



<https://embeddedmontiarc.github.io/OCL/index.html>

Thank you for your attention!