Collected Experience and Thoughts on Long Term Development of an Open Source MDE Tool

OSS4MDE 2014, Valencia, Spain

Lars Hamann, Frank Hilken, and Martin Gogolla

Universität Bremen

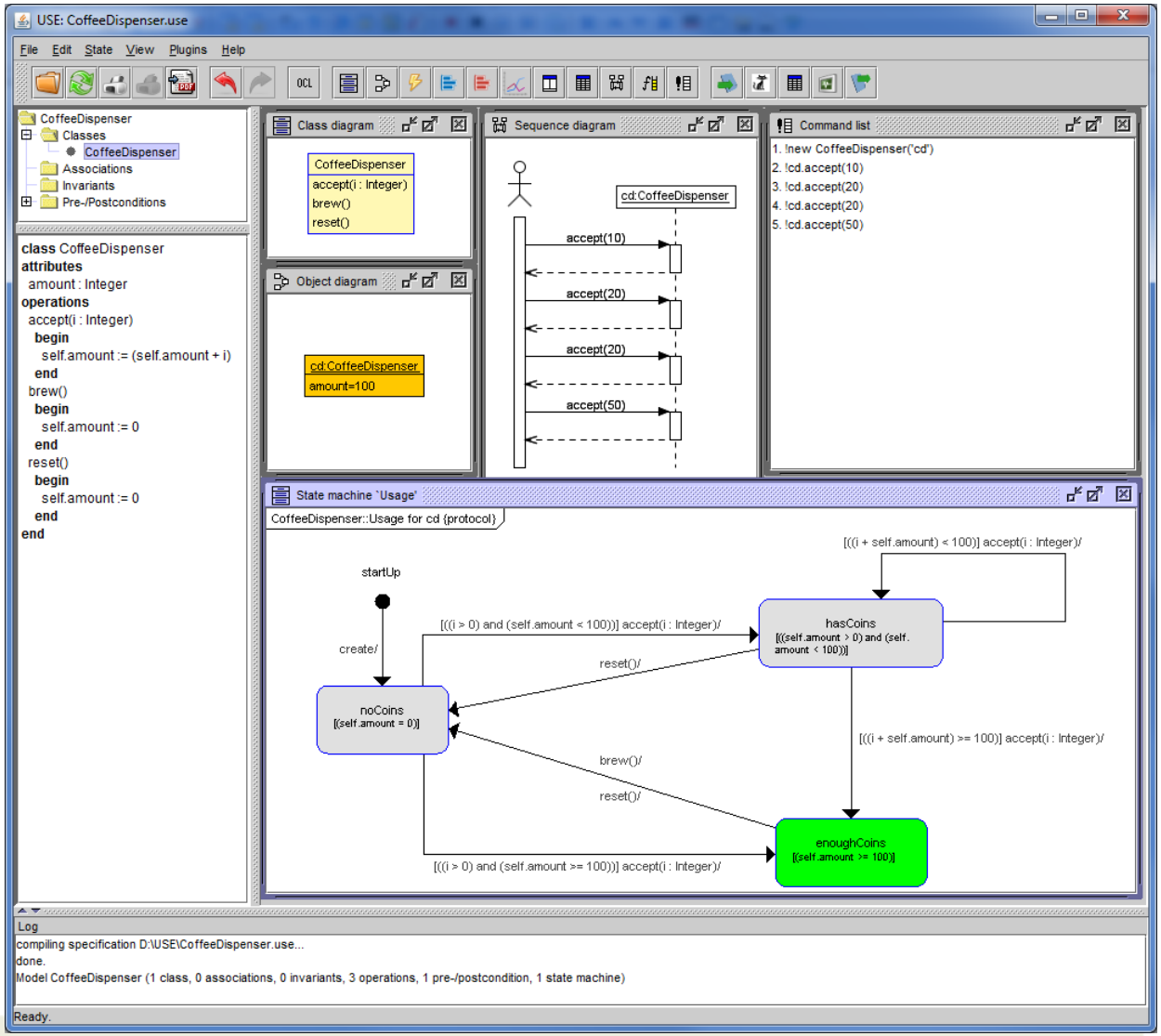
Database Systems Group
Department for Mathematics and Computer Science
University of Bremen, Germany

Overview

- USE
  - Applications
  - History
- Lessons learned
  - Factory of Success
  - Work of Students
    - Chamber of Horrors
- Conclusion

USE

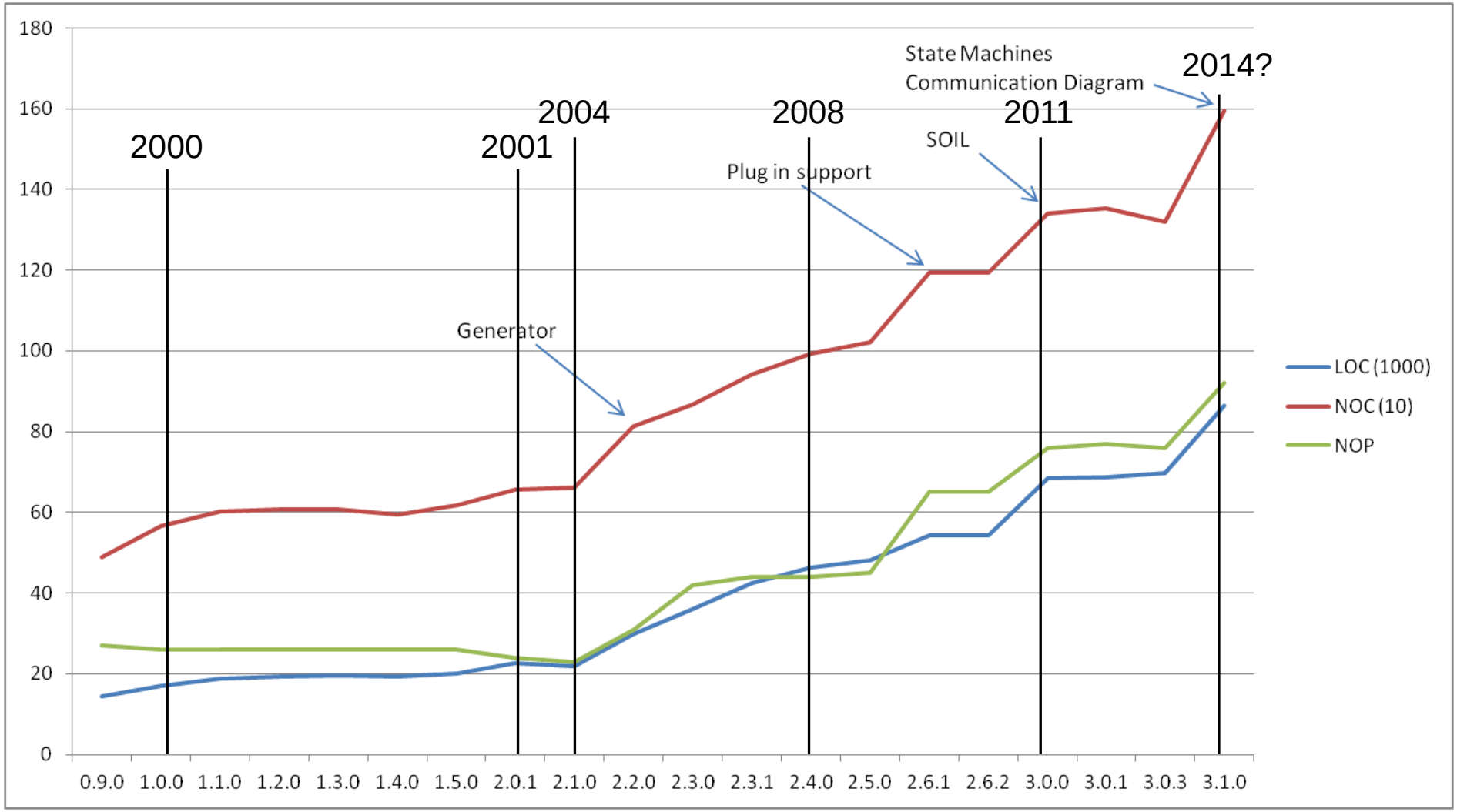
- Acronym for:
  UML-based **S**pecification **E**nvironment
- Broad reach:
  - some 4,700 downloads in 2013
  - used by research projects
  - industrial use
- Based on the JRE, only


USE: CoffeeDispenser.use
File Edit State View Plugins Help
CoffeeDispenser
Classes
CoffeeDispenser
Associations
Invariants
Pre-/Postconditions
class CoffeeDispenser
attributes
amount : Integer
operations
accept(i : Integer)
begin
self.amount := (self.amount + i)
end
brew()
begin
self.amount := 0
end
reset()
begin
self.amount := 0
end
end
Class diagram
CoffeeDispenser
accept(i : Integer)
brew()
reset()
Object diagram
cd:CoffeeDispenser
amount=100
Sequence diagram
cd:CoffeeDispenser
accept(10)
accept(20)
accept(20)
accept(50)
Command list
1. !new CoffeeDispenser('cd')
2. !cd.accept(10)
3. !cd.accept(20)
4. !cd.accept(20)
5. !cd.accept(50)
State machine 'Usage'
CoffeeDispenser::Usage for cd {protocol}
startUp
create/
noCoins
[(self.amount = 0)]
[((i > 0) and (self.amount < 100))] accept(i : Integer)/
hasCoins
[((self.amount > 0) and (self.amount < 100))]
[((i + self.amount) < 100)] accept(i : Integer)/
reset()/
[((i + self.amount) >= 100)] accept(i : Integer)/
brew()/
reset()/
enoughCoins
[(self.amount >= 100)]
[((i > 0) and (self.amount >= 100))] accept(i : Integer)/
Log
compiling specification D:\USE\CoffeeDispenser.use...
done.
Model CoffeeDispenser (1 class, 0 associations, 0 invariants, 3 operations, 1 pre-/postcondition, 1 state machine)
Ready.

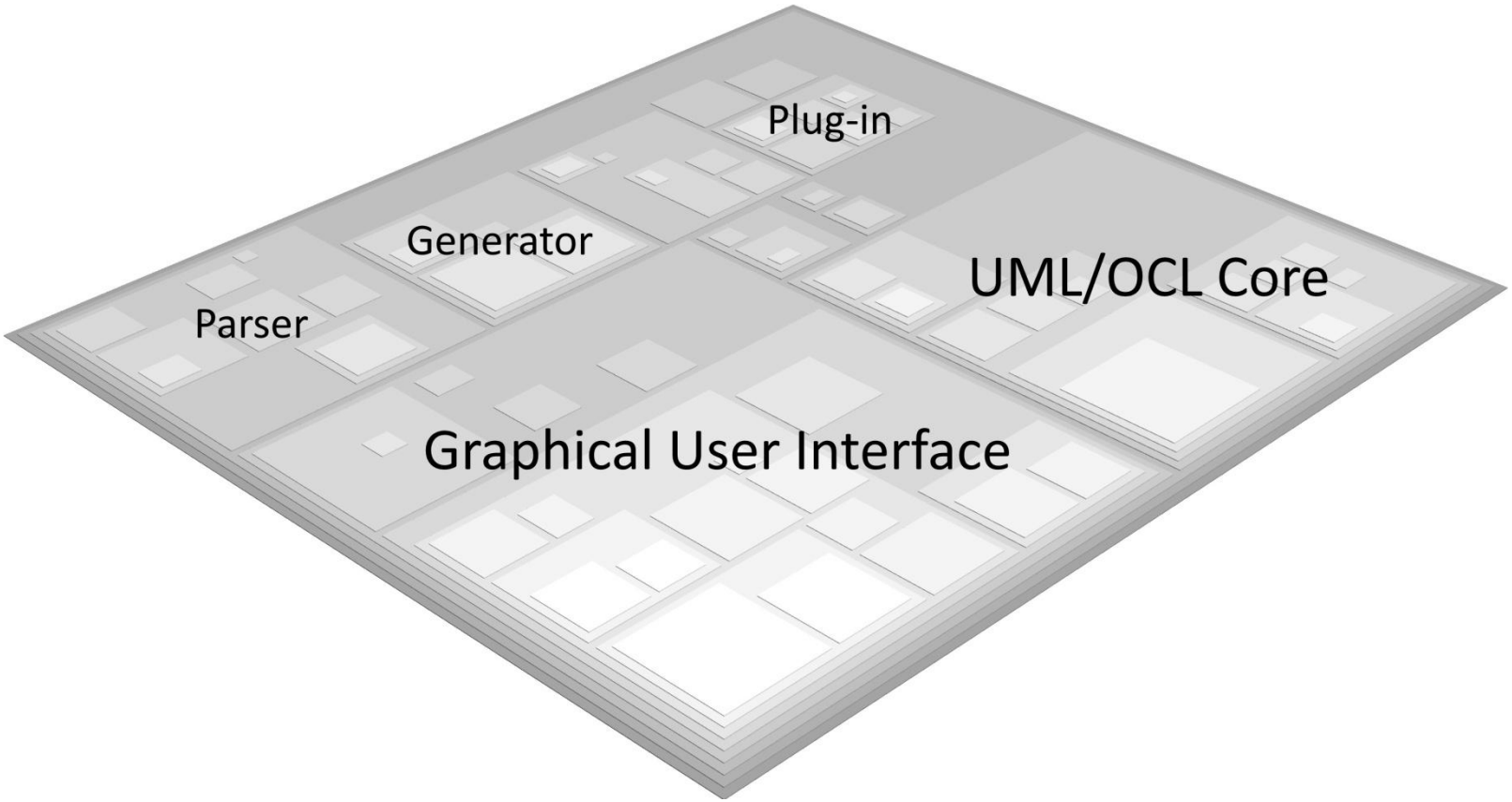
Applications of USE

- Modeling
  - Validation
  - Verification
  - Simulation
- Education
  - “Playing with models”
- Component
  - OCL-Compiler and Interpreter

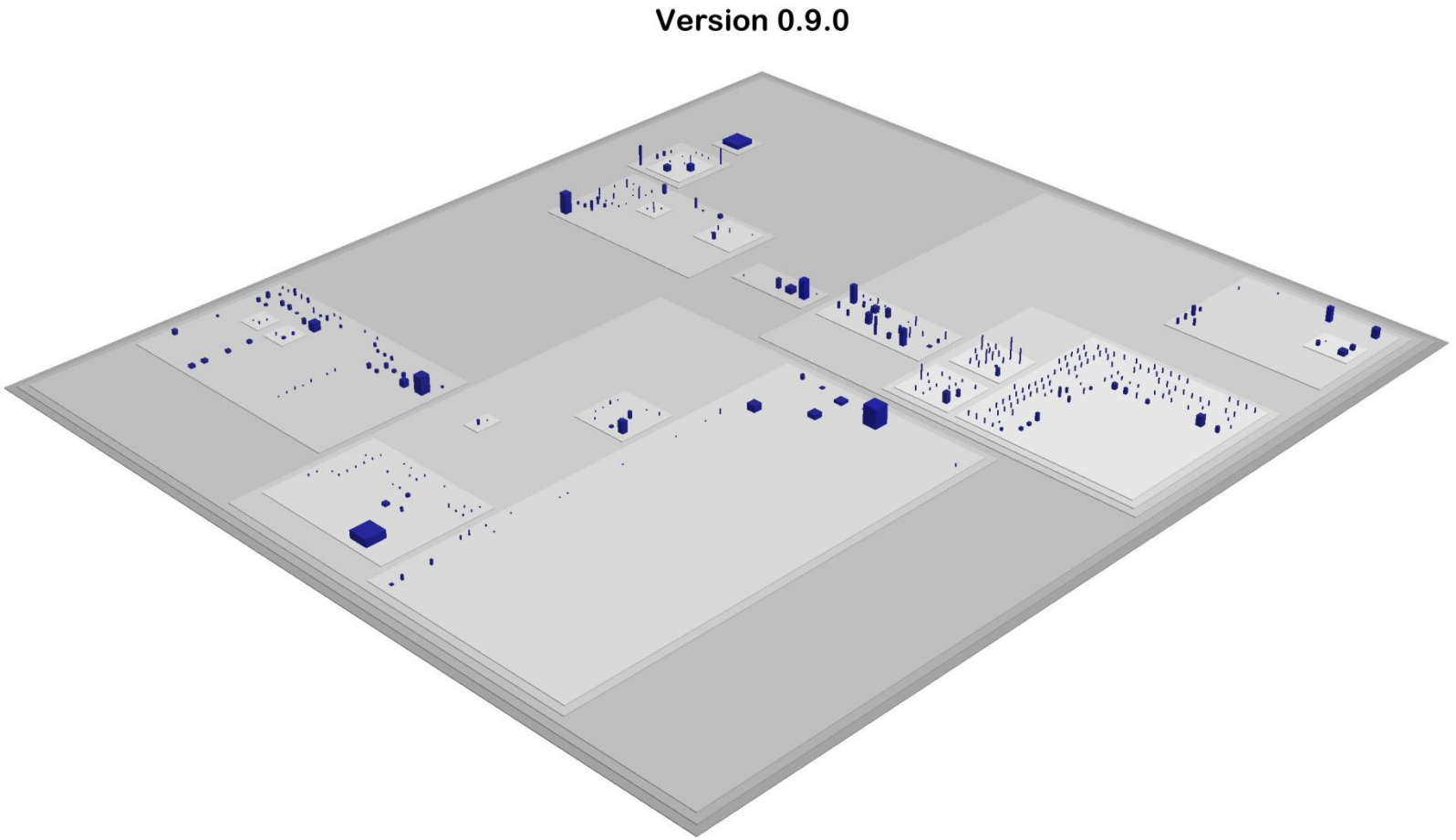
History of USE by Numbers



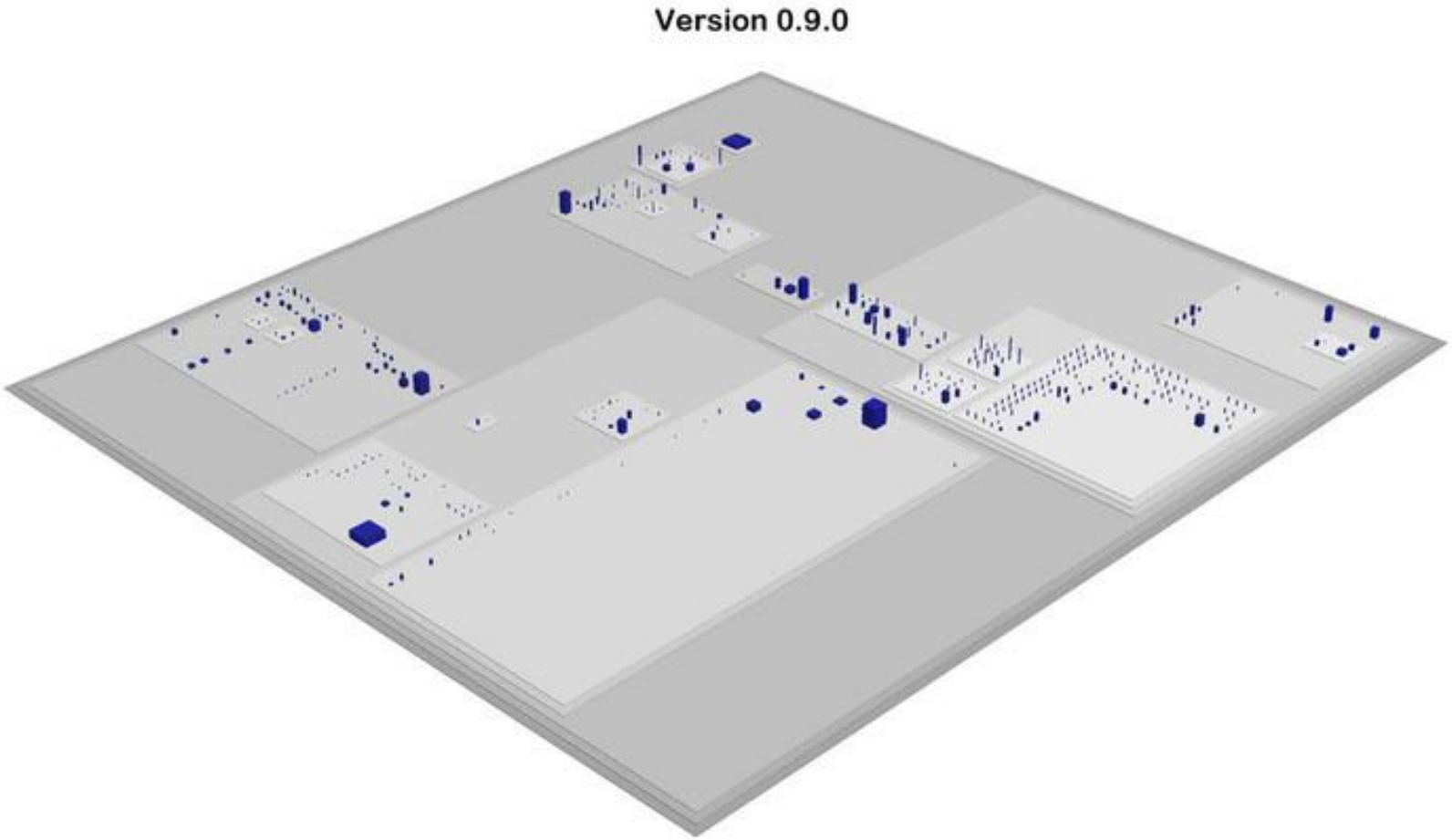
USE: a growing city



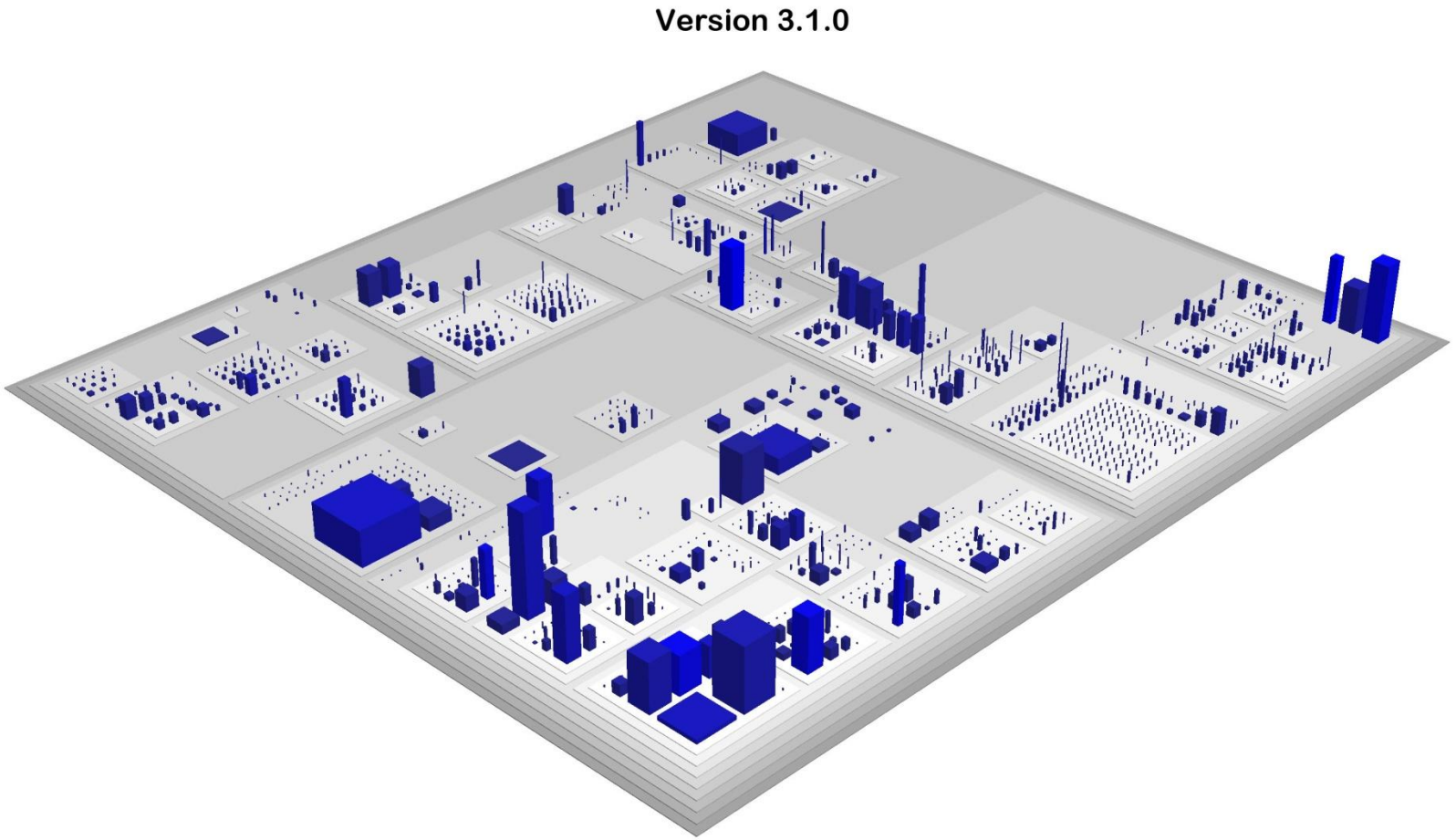
USE: a growing city



USE: a growing city



USE: a growing city



Technical

- Stable Core
- Test, test, test…
  - More than 20,000 OCL expressions
- Continuous Integration
- Team including leading „Geek“

Organizational

- Industrial Cooperations (Money…)
- How to not become a service provider?
  - We don‘t have an answer, yet

Community and Feedback

- Open Source does not provide Devs „for free“
- Publications
- Research Cooperations
- Feedback by Users, e. g., Students and Modeler
- Profound and fast(!) Support
- Students help to identify usability issues

Extensions by Students

- Quality varies heavily
- Required time, too
  - Can block interesting topics)

- How to improve quality?
  - Kick-off meeting to explain:
    - topic and
    - relevant parts of the system

Extensions by Students

- How to improve quality?
  - Best practice guide to overcome simple errors
  - Regular meetings (also reduces required time)
    - if possible: Code Review
- Not all work is going to be useful
- After Integration, if any:
  - Refactoring
  - Developer left the company!

Chamber of Horros

Controle structures?

```
 1: // compile command
 2: MSystem system = system();
 3: List<MCmd> cmdList = CMDCompiler.compileCmdList(…)
 4:
 5: // compile errors?
 6: String message = (new ShowHideExec(line)).exec();
 7: System.out.println(message);
 8:
 9: if (cmdList == null){ return; }
10:
11: for (MCmd cmd : cmdList) {
12:   …
13: }
```

Strings rule the world!

```
1: // break if the parent node is an exists node
2: // and child evaluates to "true"
3: // for minimal evaluation
4: else if (!fExtendedExists.isSelected()
5:           && parentExpr.name() == "exists"
6:           && child.getResult() == "true")
7:    break breakLabel;
```

```
1: // break if the parent node is an exists node
2: // and child evaluates to "true"
3: // for minimal evaluation
4: else if (!fExtendedExists.isSelected()
5:           && parentExpr instanceof ExpExists
6:           && child.getResult().equals(BooleanValue.TRUE))
7:    break breakLabel;
```

Who needs the M in MVC?

```
 1: for (int i = 0; i < fAttributes.size(); i++) {
 2: // Class is shown in a table, e.g., „Person (1-7)“
 3: String cname =
 4:  fAttributes.get(i).toString().substring(0,
 5:  fAttributes.get(i).toString().indexOf("(")).trim();
 6: Iterator it = selectedClasses.iterator();
 7: // find out, which mclass selected
 8: while (it.hasNext()) {
 9:   MClass mc = (MClass) (it.next());
10:   if (mc.name().equals(cname))
11:     break;
12: }
13: …
```

Why reuse?

```
MMultiplicity mult1 = null, mult2 = null;

// for many
if (multiplicity1 == "0..*")
  mult1 = new MMultiplicity(0, -1);
// zero to one
else if (multiplicity1 == "0..1")
  mult1 = new MMultiplicity(0, 1);
…
if (multiplicity2 == "0..*")
  mult1 = new MMultiplicity(0, -1);
else if (multiplicity2 == "0..1")
  mult1 = new MMultiplicity(0, 1);
```

Summary

- Stability over Features
  - Continuous Integration
  - Test
- Cooperation
- Students need assistance
- Integration of work of students usually not “as-is”

USE can be found here:
http://www.sourceforge.net/projects/useocl/
or google “use ocl bremen”

Thank you for your attention!

Any questions or remarks?