

Fast, integrated and debuggable Interpreters in MPS and beyond

Niko Stotz



LangDev Meetup 2022

What and why: Interpreters

- Execute user programs immediately
- Interactive responses → Excel
- Easy to implement

```
TrafficLight_Test2 x
test for TrafficLight
TrafficLight_Test2 {
  assert state red
  trigger pedestrianButton
  assert state error
}

TrafficLight x
state machine TrafficLight {
  @Requirements for Traffic Light
  event pedestrianButton
  event evt2
  event evt3
  var decimal = 0.0003
  var counter = 3
  var name = "test"
  var run = true
  @Requirements for Traffic Light
  initial state red {
    on pedestrianButton [counter == 3 && run] -> green
    on pedestrianButton [counter == 0] -> error
    on evt2 [decimal == 0.0003] -> green
    on evt3 [name == "test"] -> error
  }
  @Requirements for Traffic Light
  state green {

TrafficLight_Test1 x
test for TrafficLight
TrafficLight_Test1 {
  assert state red
  trigger pedestrianButton
  assert state green
  trigger pedestrianButton
  assert state red
  trigger evt2
  assert state green
  trigger evt2
  assert state green
  trigger evt3
  assert state red
  trigger evt3
  assert state error
  trigger evt2
  assert state red
  trigger evt3
  assert state error
  trigger evt3
  assert state green
}
```

What and why: Truffle

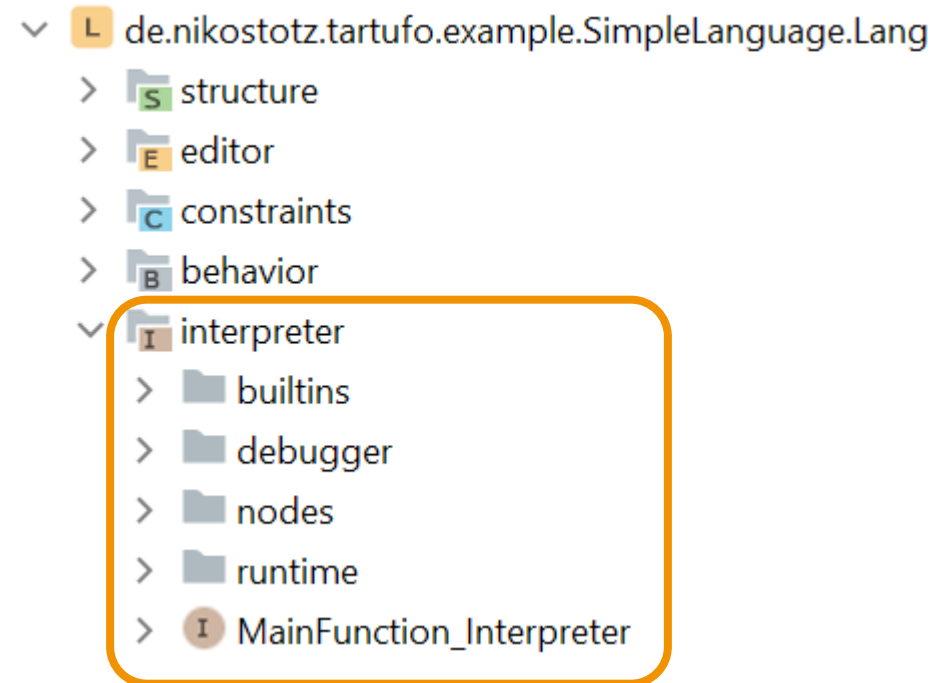
- Part of Oracle GraalVM
- Framework for high performance interpreters
- Community edition available

- Runs on any JVM
- Lots of tooling



What and why: Tartufo – Separate DSL and aspect in MPS

- New aspect in MPS language
- Lots of boilerplate
- Integral language part: defines semantic
- Ease of use



Demo



Demo screenshot: Asynchronous live update on change

```
main ( << ... >> )
{
  num = 6153
  if ( num < 1 )
  {
    return 0
  }
  else
  <no else>
  n1 = 0
  n2 = 1
  i = other ( 1 )
  while ( i < num )
  {
    next = n2 + n1
    n1 = n2
    n2 = next
    i = i + 1
  }
  return "Fibonacci of " + num + " is " + n2
}
```

Fibonacci of 6153 is 75781698331363642370210122020104031707754130042129488246182627386793034475789023201726287849278

Demo screenshot: Breakpoint in user program with variable inspector

```
main ( << ... >> )
{
  num = 6152
  if ( num < 1 )
  {
    return 0
  }
  else
  <no else>
  n1 = 0
  n2 = 1
  i = other ( 1 )
  while ( i < num )
  {
    next = n2 + n1
    n1 = n2
    n2 = next
    i = i + 1
  }
  return "Fibonacci of " + num + " is " + n2
}
/
```

Debug: T. SomeNode main ×

Debugger Console

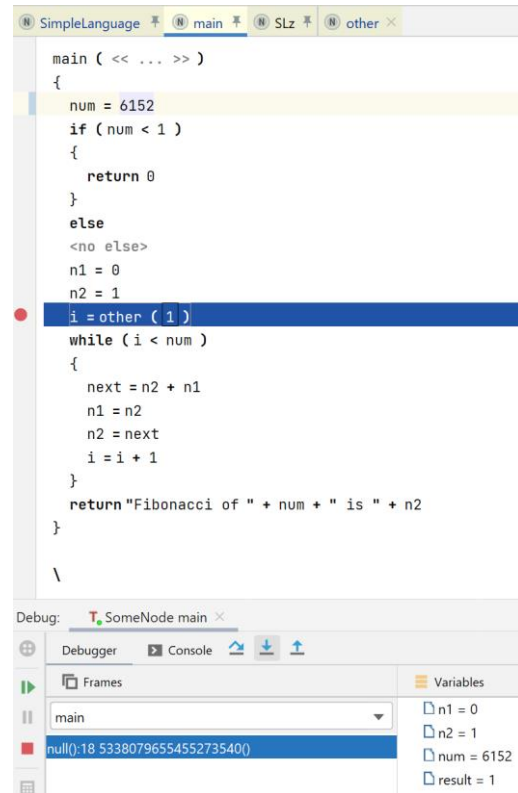
Frames

main

Variables

- n1 = 0
- n2 = 1
- num = 6152

Demo screenshot: Stepping into function



```
SimpleLanguage main SLz other x
main ( << ... >> )
{
  num = 6152
  if ( num < 1 )
  {
    return 0
  }
  else
  <no else>
  n1 = 0
  n2 = 1
  i = other ( 1 )
  while ( i < num )
  {
    next = n2 + n1
    n1 = n2
    n2 = next
    i = i + 1
  }
  return "Fibonacci of " + num + " is " + n2
}
\

Debug: SomeNode main x
Debugger Console
Frames
main
null():18 5338079655455273540()
Variables
n1 = 0
n2 = 1
num = 6152
result = 1
```


Demo screenshot: Stepping out of function

The screenshot shows a code editor with the following code:

```
other ( a )  
{  
  i = 8 + a  
  return i  
}
```

A yellow highlight is on the line `i = 8 + a`. Below the code, a message reads: "Type error at other line 5 col 1: operation "+" no".

The debugger window below shows the "Frames" pane with the following stack:

- other
- main:9:1118180674890109063()
- main:17:5338079655455273540()

The "other" frame is selected, and a "Step Out" button (Shift+F8) is visible. To the right, the "Variables" pane shows:

- a = 1
- i = 9

Demo screenshot: Export model with user program

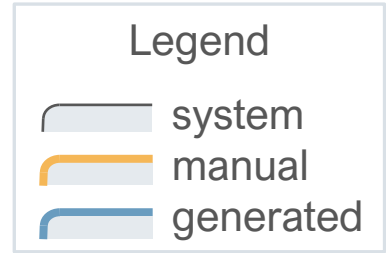
```
main ( << ... >> )
{
  num = 6153
  if ( num < 1 )
  {
    return 0
  }
  else
  <no else>
  n1 = 0
  n2 = 1
  i = other ( 1 )
  while ( i < num )
  {
    next = n2 + n1
    n1 = n2
    n2 = next
    i = i + 1
  }
  return "Fibonacci of " + num + " is " + n2
}
```

Fibonacci of 6153 is 7578169833136364237021012202010403170775413004212948824618262731

```
sole: Console
Type an expression or {statements} to execute.
Type ? for a list of commands.
Press Ctrl+Enter to execute command.
Use Ctrl+M, Ctrl+R and Ctrl+L to add imports and languages.
> node<> input = nodeRef@23968;

MpsNodeSerializer serializer = new MpsNodeSerializer();
serializer.serialize(input);
NodePersistor persistor = new NodePersistor(Path.of("C:/temp/nodeOutput.dat"));
persistor.save(serializer.getRootNodes());
```


MPS M3 representation



MPS

M3
meta meta model

Concept Definition

M2
meta model

BaseConcept

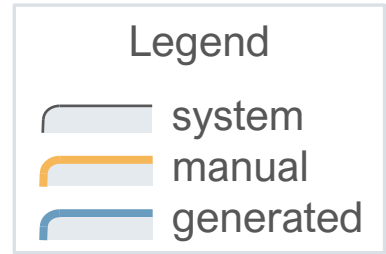
M1
model

VariableDefinition

M0
real thing

int size

Truffle M3 representation



MPS

Truffle

M3
meta meta model

Concept Definition

Class

M2
meta model

BaseConcept

Truffle Node

M1
model

VariableDefinition

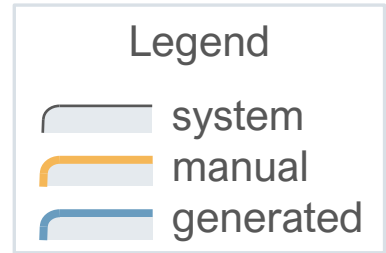
VariableNode

M0
real thing

int size

size

Types of M3 elements



MPS

«ConceptDefinition»

Concept Definition

«ConceptDefinition»

BaseConcept

«ConceptDefinition»

VariableDefinition

«Node»

int size

Truffle

«Class»

Class

«Class»

Truffle Node

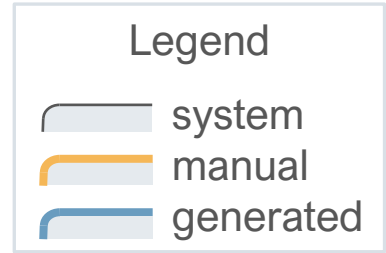
«Class»

VariableNode

«Object»

size

No MPS in Truffle realm



all MPS

MPS

all Java
no MPS

Truffle

«ConceptDefinition»

Concept Definition

«Class»

Class

«ConceptDefinition»

BaseConcept

«Class»

Truffle Node

«ConceptDefinition»

VariableDefinition

«Class»

VariableNode

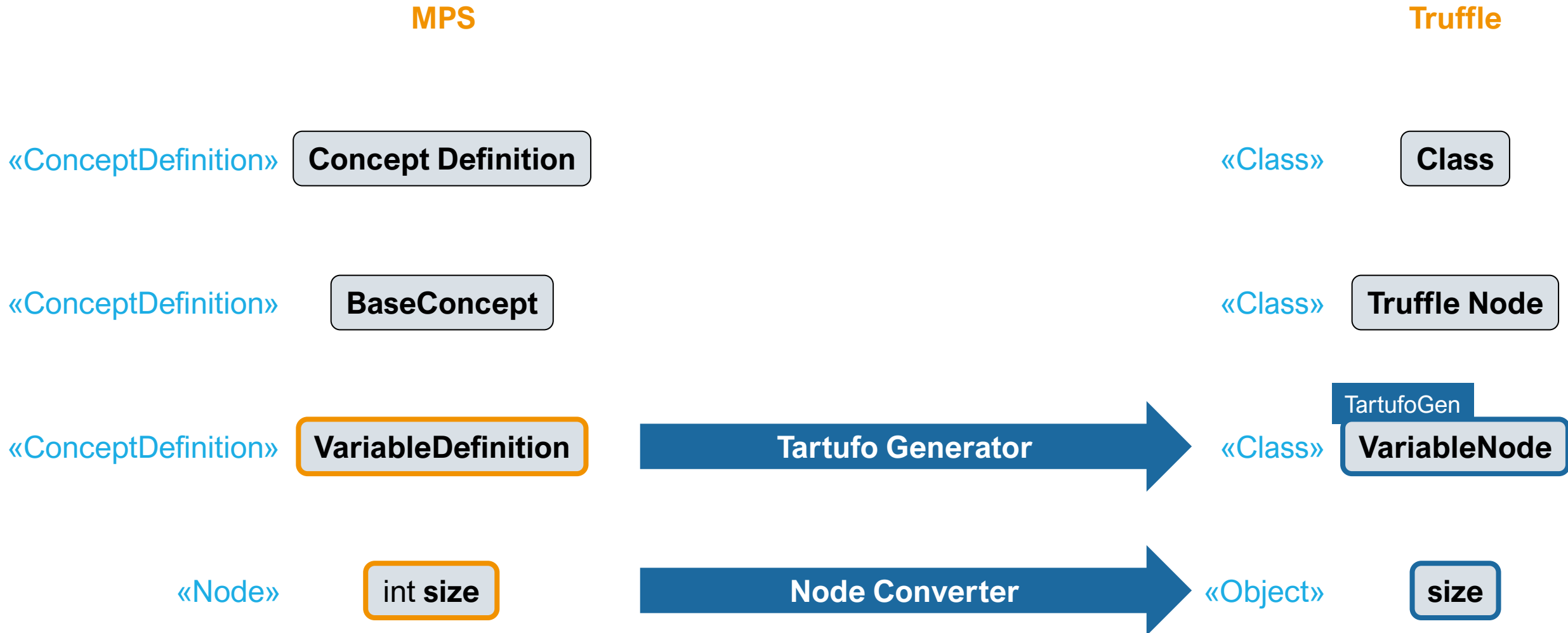
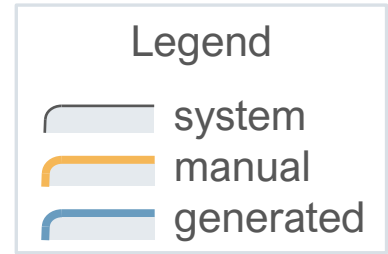
«Node»

int size

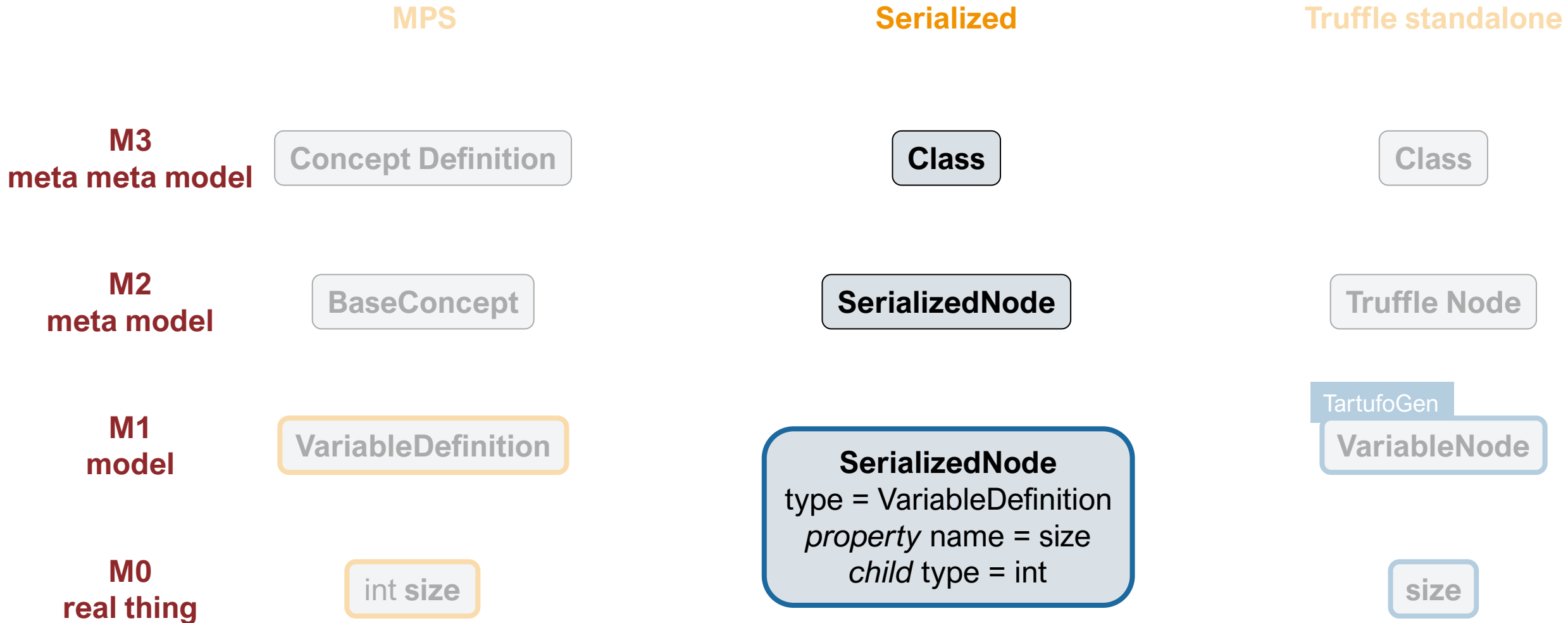
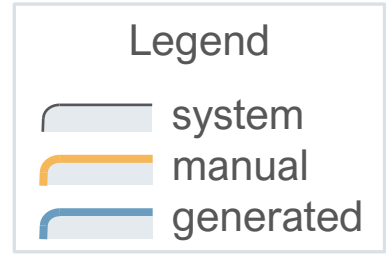
«Object»

size

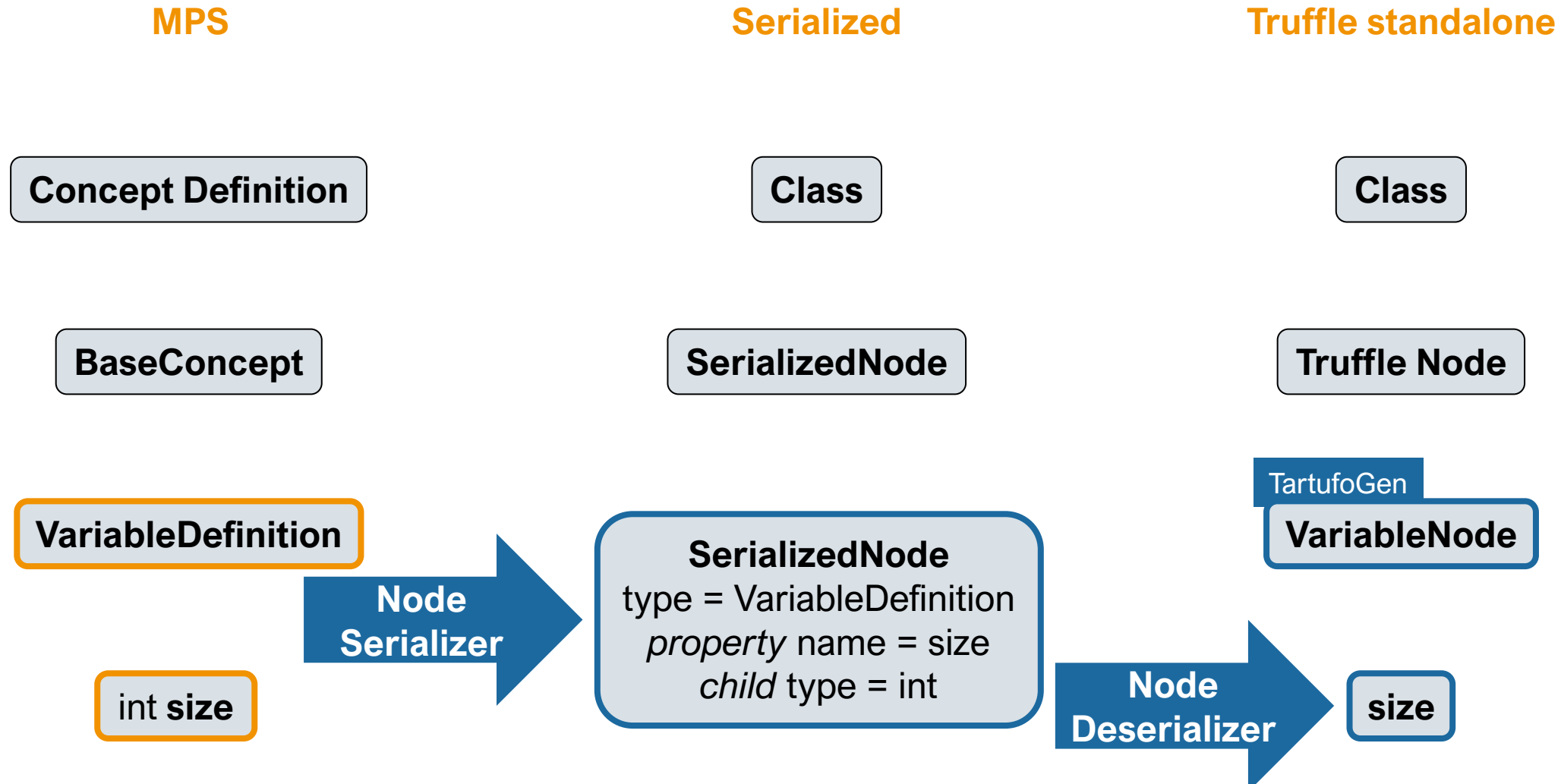
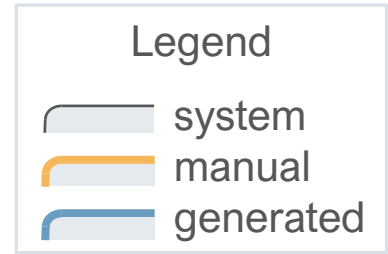
Manual MPS elements generated to Truffle realm



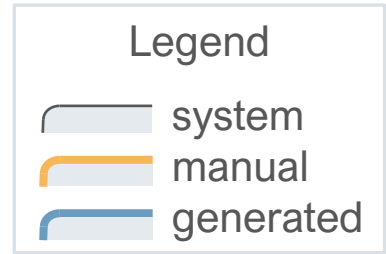
Custom serialization M3 representation



MPS serialization, deserialization to Truffle



Types of M3 elements



MPS

«ConceptDefinition»

Concept Definition

«ConceptDefinition»

BaseConcept

«ConceptDefinition»

VariableDefinition

«Node»

int size

Serialized

«Class»

Class

«Class»

SerializedNode

«Object»

SerializedNode
type = VariableDefinition
property name = size
child type = int

Truffle standalone

«Class»

Class

«Class»

Truffle Node

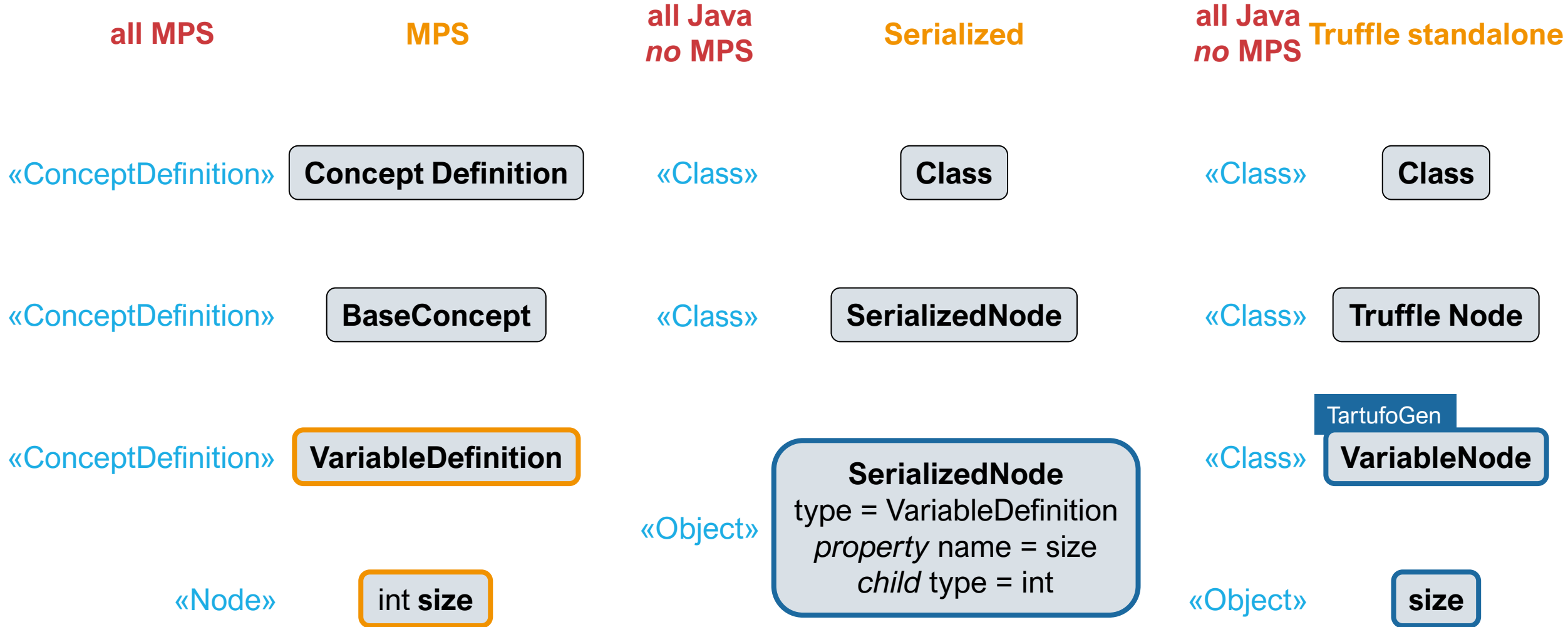
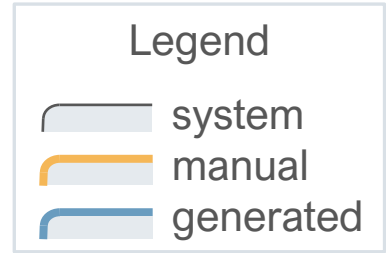
«Class»

TartufoGen
VariableNode

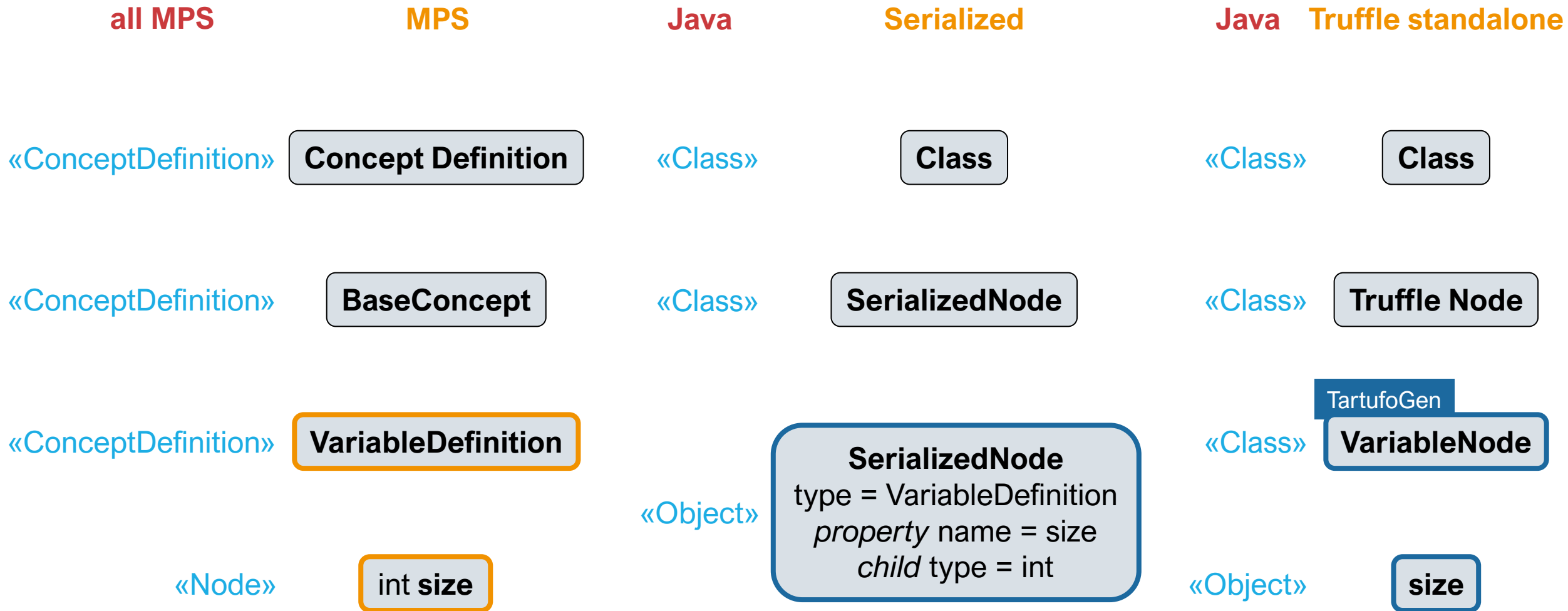
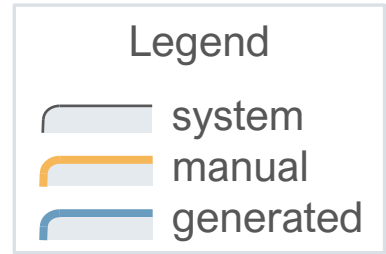
«Object»

size

No MPS in serialization or Truffle realms



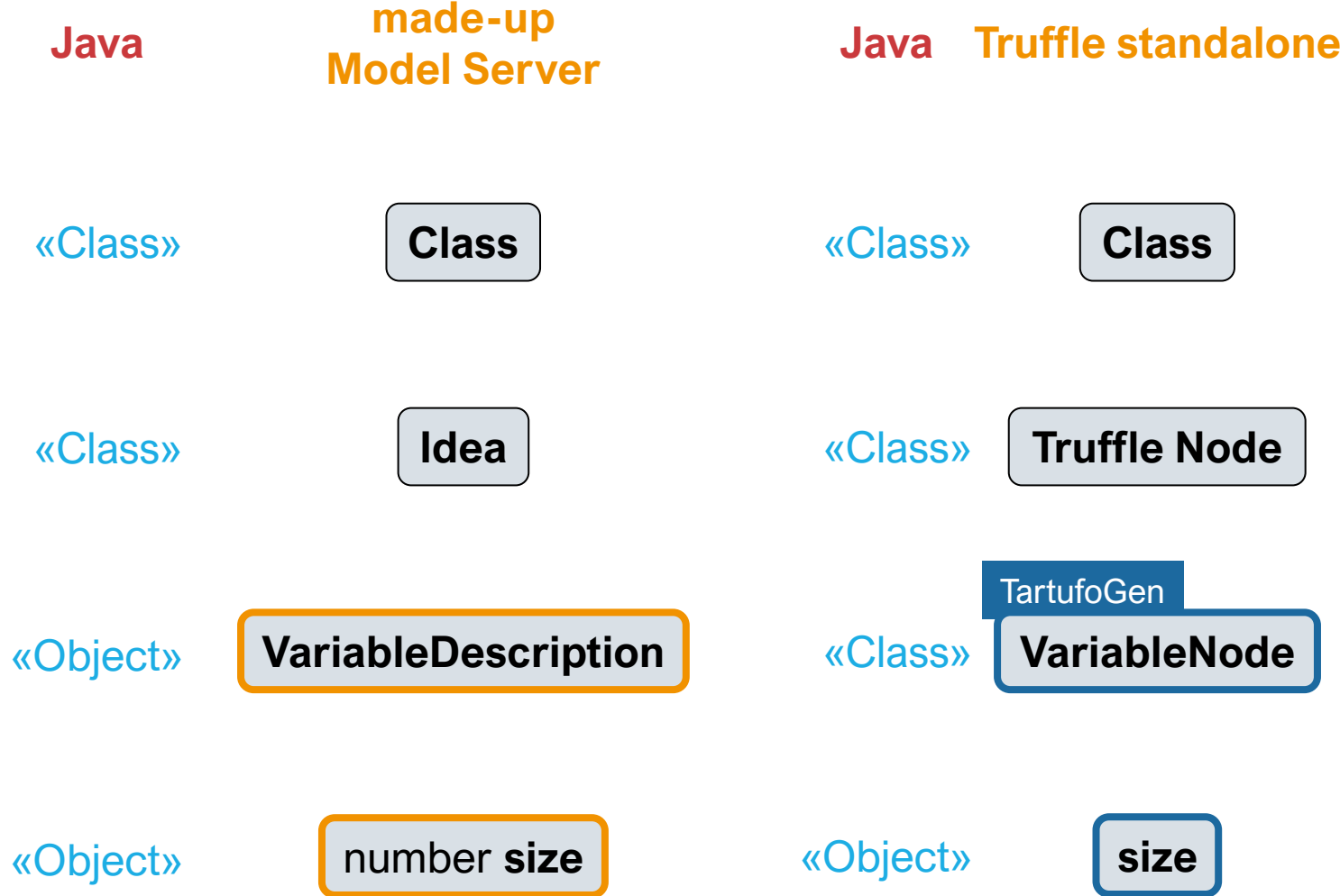
Focus on Java part – without MPS

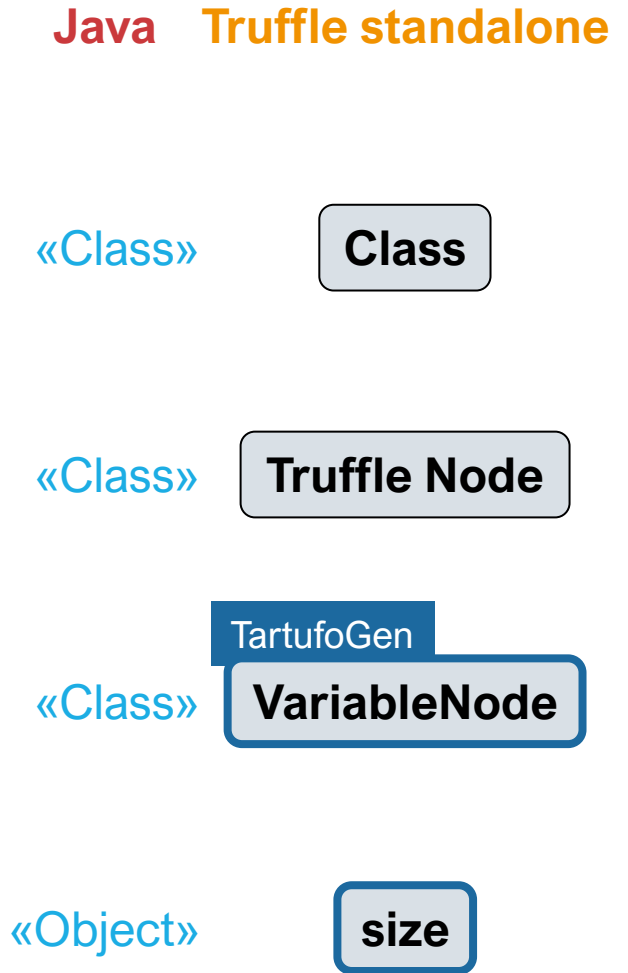
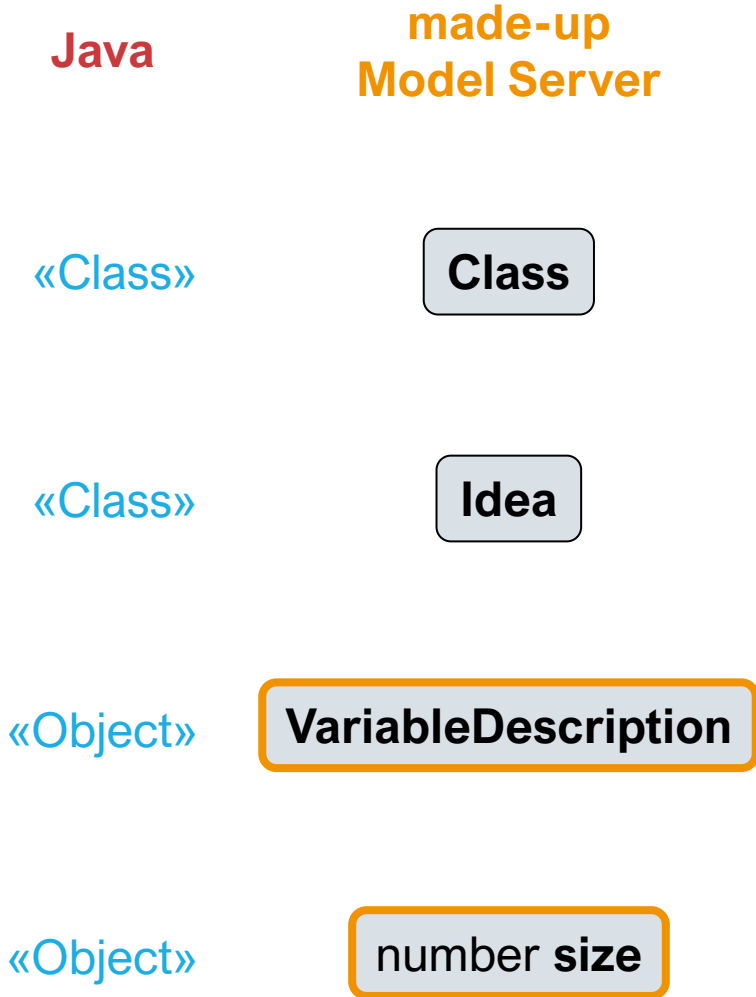
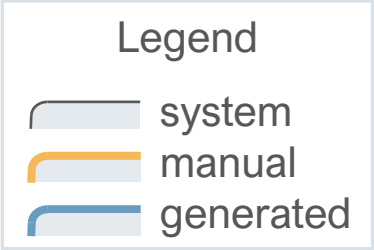


Serialization could also be model server

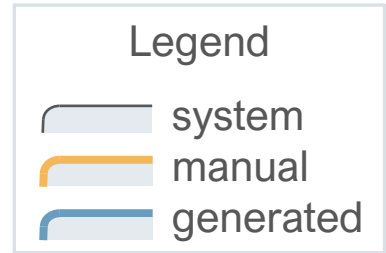
Legend

- system
- manual
- generated

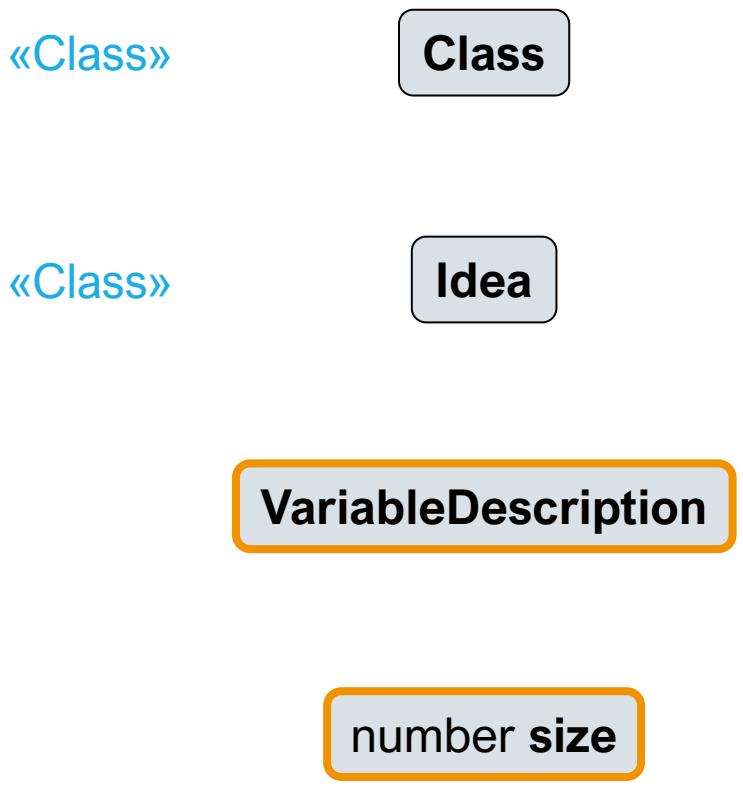




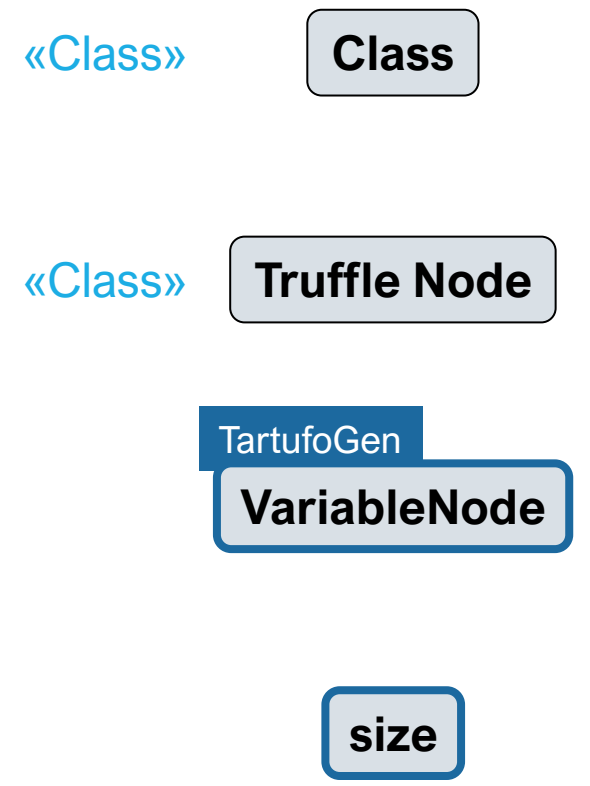
Model server to Truffle converter



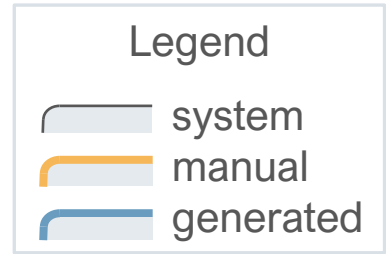
made-up
Model Server



Truffle standalone



API call instead of node converter



made-up
Model Server

Truffle
Interpretation Server

VariableDescription

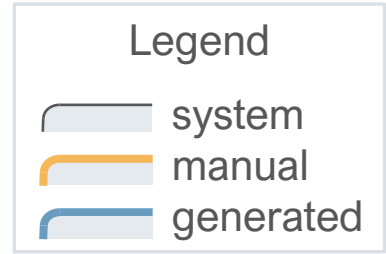
TartufoGen
VariableNode

number size



size

Model server in any language

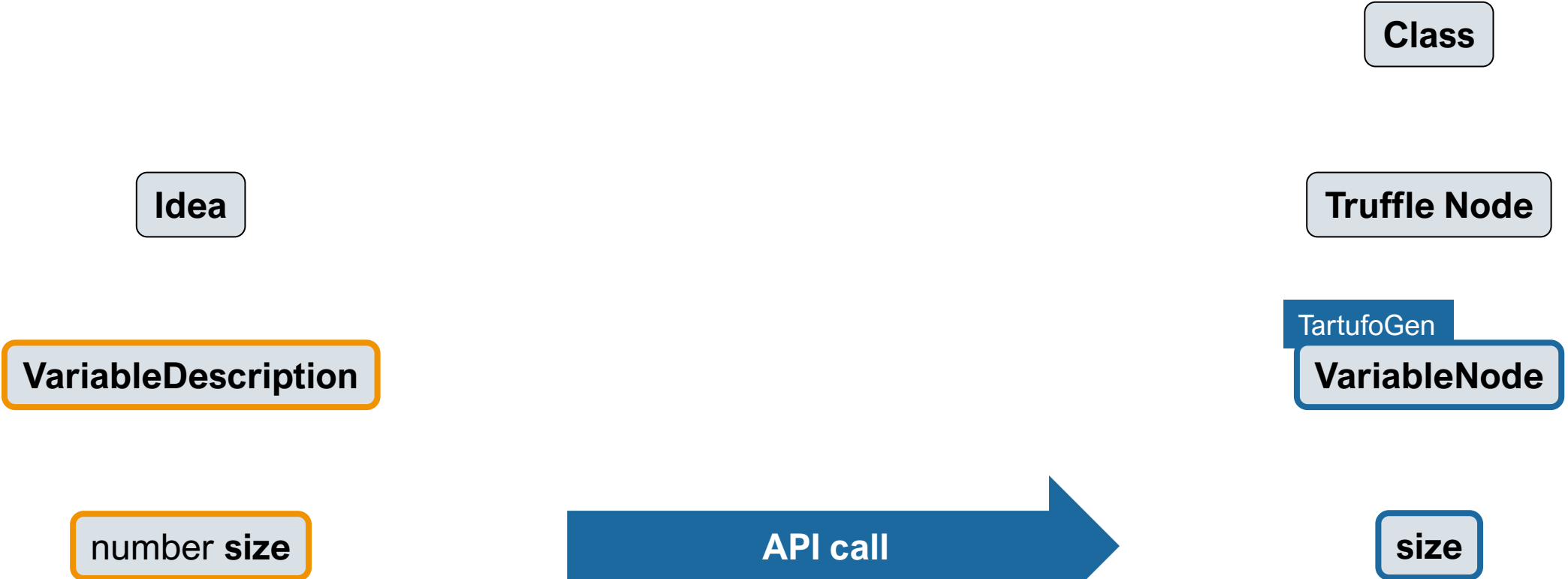


any language

made-up Model Server

Java

Truffle Interpretation Server



Tartufo Languages

Implementation

- **Tartufo Interpreter**
 - Tartufo Abstraction?
- Tartufo Base
- Tartufo Mixin
- Truffle Annotations
- Java
- Byte code

Infrastructure

- ~~Converter~~
- Service
- Dummyclass
- Aspect
- **BLIntegration**

Language: Tartufo Interpreter

```
Interpreter SimpleLanguage
```

```
Evaluators
```

```
AddExpression specialized
```

```
exception on ArithmeticException
```

```
long lhs, long rhs {
```

```
    Math.addExact(lhs, rhs);
```

```
}
```

```
type BigInteger lhs, BigInteger rhs {
```

```
    lhs.add(rhs);
```

```
}
```

```
guard Object lhs, Object rhs
```

```
if lhs instanceof string || rhs instanceof string; {
```

```
    lhs.toString() + " " + rhs.toString();
```

```
}
```

Language: Tartufo Base

```
public abstract Specialized Node Add extends Binary {
  Description          <no description>
  Short Name           +
  Generate Wrapper    false
  Refers to Type System false
  Tags                << ... >>
  Implemented Libraries << ... >>
  Shared Caches       << ... >>
  Frame Kind          <no frameKind>
  Node Children       << ... >>

  protected string add
  (
    specialized Object left
    specialized Object right
  )
  @Specialization ( guard return left instanceof string || right instanceof string; )
  {
    return left.toString() + " " + right.toString();
  }
}
```

Language: Tartufo Mixin

```
@NodeInfo(shortName = "+")
public abstract class SLAddNode extends SLBinaryNode {
    @Specialization(rewriteOn = ArithmeticException.class)
    protected long add(long left, long right) {
        return Math.addExact(left, right);
    }
    @Specialization()
    @TruffleBoundary()
    protected SLBigNumber add(SLBigNumber left, SLBigNumber right) {
        return new SLBigNumber(left.getValue().add(right.getValue()));
    }
    @Specialization(guards = isString(left, right))
    @TruffleBoundary()
    protected String add(Object left, Object right) {
        return left.toString() + " " + right.toString();
    }
    protected boolean isString(Object a, Object b) {
        return a instanceof String || b instanceof String;
    }
    @Fallback
    protected Object typeError(Object left, Object right) {
        throw SLException.typeError(this, left, right);
    }
}
```

Language: Truffle Annotations

```
@NodeInfo(shortName = "+")
public abstract class SLAddNode extends SLBinaryNode {
    @Specialization(rewriteOn = ArithmeticException.class)
    protected long add(long left, long right) {
        return Math.addExact(left, right);
    }
    @Specialization
    @CompilerDirectives.TruffleBoundary
    protected SLBigNumber add(SLBigNumber left, SLBigNumber right) {
        return new SLBigNumber(left.getValue().add(right.getValue()));
    }
    @Specialization(guards = "isString(left, right)")
    @CompilerDirectives.TruffleBoundary
    protected String add(Object left, Object right) {
        return left.toString() + " " + right.toString();
    }
    protected boolean isString(Object a, Object b) {
        return a instanceof String || b instanceof String;
    }
    @Fallback
    protected Object typeError(Object left, Object right) {
        throw SLException.typeError(operation: this, left, right);
    }
}
```

Language: Java

@GeneratedBy(SLAddNode.class)

```
public final class SLAddNodeGen extends SLAddNode {
```

1 usage

```
private SLAddNodeGen(SLExpressionNode leftNode, SLExpressionNode rightNode) {  
    this.leftNode_ = leftNode;  
    this.rightNode_ = rightNode;  
}
```

1 usage

```
private Object executeGeneric_generic1(int state_0, VirtualFrame frameValue) {  
    Object leftNodeValue_ = this.leftNode_.executeGeneric(frameValue);  
    Object rightNodeValue_ = this.rightNode_.executeGeneric(frameValue);  
    if ((state_0 & 0b1) != 0 /* is-state_0 add(long, long) */ && leftNodeValue_ instanceof Long) {...}  
    if ((state_0 & 0b10) != 0 /* is-state_0 add(SLBigNumber, SLBigNumber) */ && SLTypesGen.isImplicitSLBigNumber((state_0 & 0b110000) >>> 4 /*  
        SLBigNumber leftNodeValue__ = SLTypesGen.asImplicitSLBigNumber((state_0 & 0b110000) >>> 4 /* extract-implicit-state_0 0:SLBigNumber */,  
        if (SLTypesGen.isImplicitSLBigNumber((state_0 & 0b11000000) >>> 6 /* extract-implicit-state_0 1:SLBigNumber */, rightNodeValue_)) {  
            SLBigNumber rightNodeValue__ = SLTypesGen.asImplicitSLBigNumber((state_0 & 0b11000000) >>> 6 /* extract-implicit-state_0 1:SLBigNum  
            return add(leftNodeValue__, rightNodeValue__);  
        }  
    }  
}  
if ((state_0 & 0b1100) != 0 /* is-state_0 add(Object, Object) || typeError(Object, Object) */) {  
    if ((state_0 & 0b100) != 0 /* is-state_0 add(Object, Object) */) {  
        if ((isString(leftNodeValue_, rightNodeValue_))) {  
            return add(leftNodeValue_, rightNodeValue_);  
        }  
    }  
}
```


Language: Base Language integration

```
Object interpretationResult = interpret ( node , editorContext.getRepository() );
```

```
CompletableFuture<Object> interpretationFuture = interpretAsync ( node , editorContext.getRepository() );
```

Maturity

MPS Truffle integration

engineered, no production

Async editor cell

prototype, usable base

Debugger

prototype, usable base

Node (de)serializer

proof of concept

Standalone executor

proof of concept

Languages

Interpreter

design ideas

Base

concepts almost complete, generator missing

Mixin

engineered, no production

Issues

- MPS Java facet vs. Java Annotation Processors
- Line numbers
- Language ease vs. extensibility

Outlook

- Remote debugging
- Debugging interaction, e.g. change values
- Node replacement
- Higher level languages
- Async editor cell as separate language
- Update to current MPS / Truffle versions
 - used: MPS 2021.1 / GraalVM 21.1
 - current: MPS 2021.3 / GraalVM 22.2

Future

- Typesystem integration
- Interop objects language

Summary

- Interpreters useful for DSLs
- Truffle provides speed and tooling
- Tartufo simplifies usage
- Enables use cases inside and outside MPS



niko@f1re.io

Addendum: Links

- Jobs @ F1RE: <https://www.f1re.nl/alle-vacatures>
- GraalVM: <https://www.graalvm.org/>
- Truffle: <https://www.graalvm.org/22.2/graalvm-as-a-platform/language-implementation-framework/>
- Tartufo will be Open Source, we first need to sort out some licensing details. Contact niko@f1re.io for early access.