Kermeta

in compiled mode

Cyril Faucher

IRISA Lab / INRIA Rennes, France
Triskell Group

Kermeta Day - April 2nd, 2009

Outline

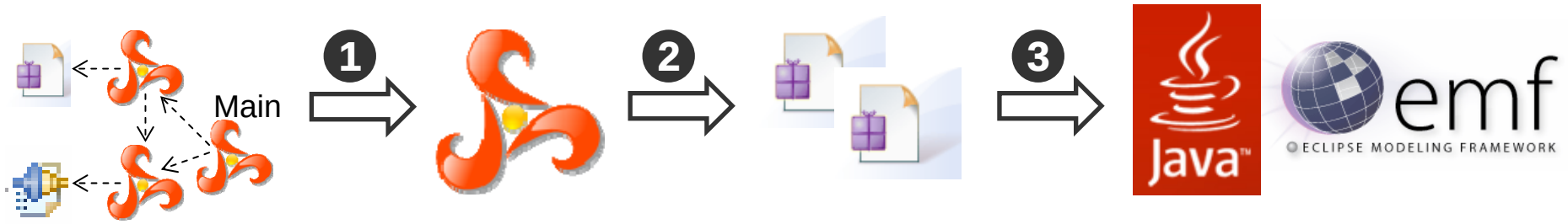
- Motivation
- Compilation process: Kmt to Java/EMF plugin
- Need of a model to complement an *.ecore: Simk
- How to customize a compilation process
- Implementation details
- Experiments
- Conclusion

Motivation

- Why a compiler ?
  - improve the execution performance
- Why Java/EMF ?
  - more deployable: Eclipse Plugin, Java Standalone
  - interoperability with the others MDE tools
  - the Kermeta interpreted mode and tooling are based on EMF, written in Java and integrated in Eclipse
  - ...

Compilation process

- A compilation process is executed
  - in **Eclipse**
  - by a right-click on the **main** Kmt of the Kermeta program
- 1rst step: **merge** the main Kmt dependencies, i.e. a Km file containing all the required resources
- 2nd step: transformation of the **Km** merged to **Ecore + Simk**
- 3rd step: plugin generation with **Java/EMF code sources** based on the **EMF Jet templates**



Metamodel as *.ecore and as Plugin and *.kmts

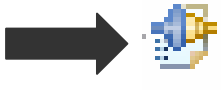
*.km merged with all the required resources

*.ecore with annotations

*.simk

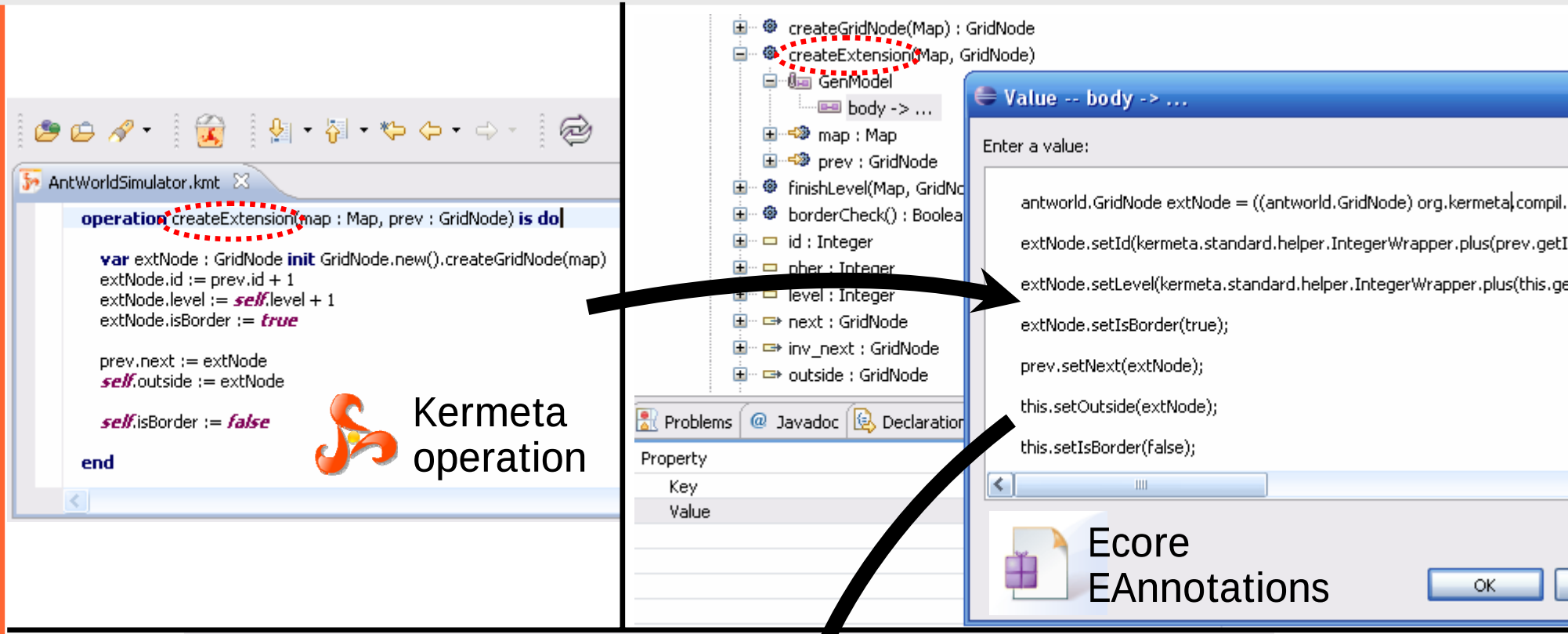
EMF plugin with Java sources

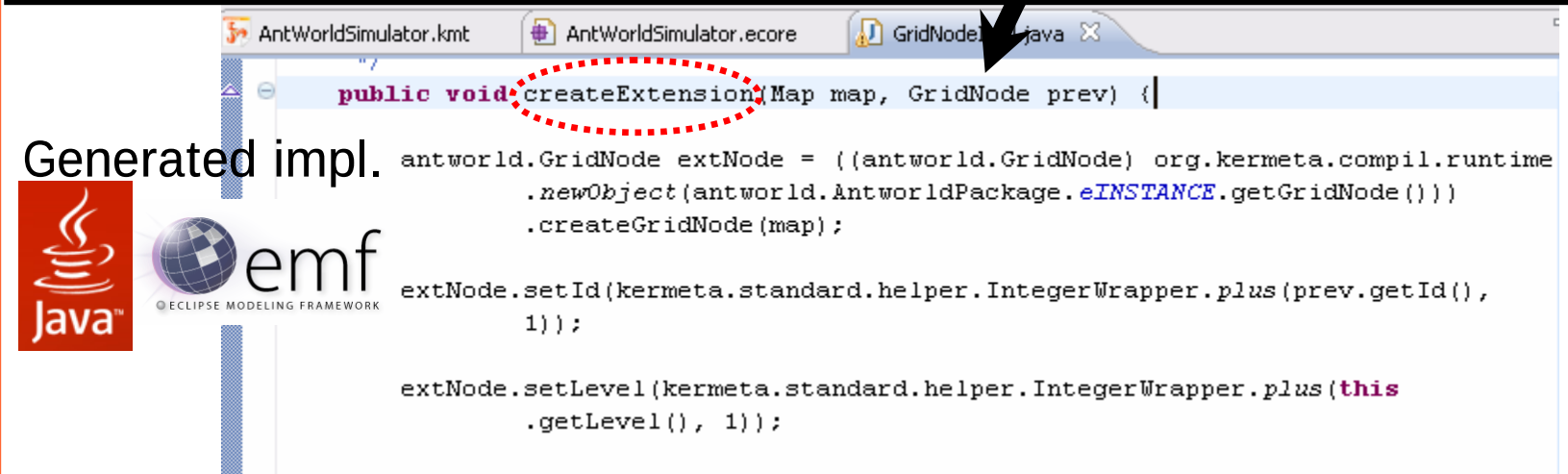
Compilation process

- **Output**  a plugin containing
  - *classical EMF model* Java Classes (interface + impl.)
  - Kermeta **behavior** in Java
  - Java **Main** methods generated to ease the launch of Java application (packages “runner”)
  - **helpers** and **extern** impls. dedicated to the Kermeta framework
  - a copy of the *.km merged for the **reflection**

=> All the resources are present in the generated plugin 

Compilation process





Generated impl.

Java

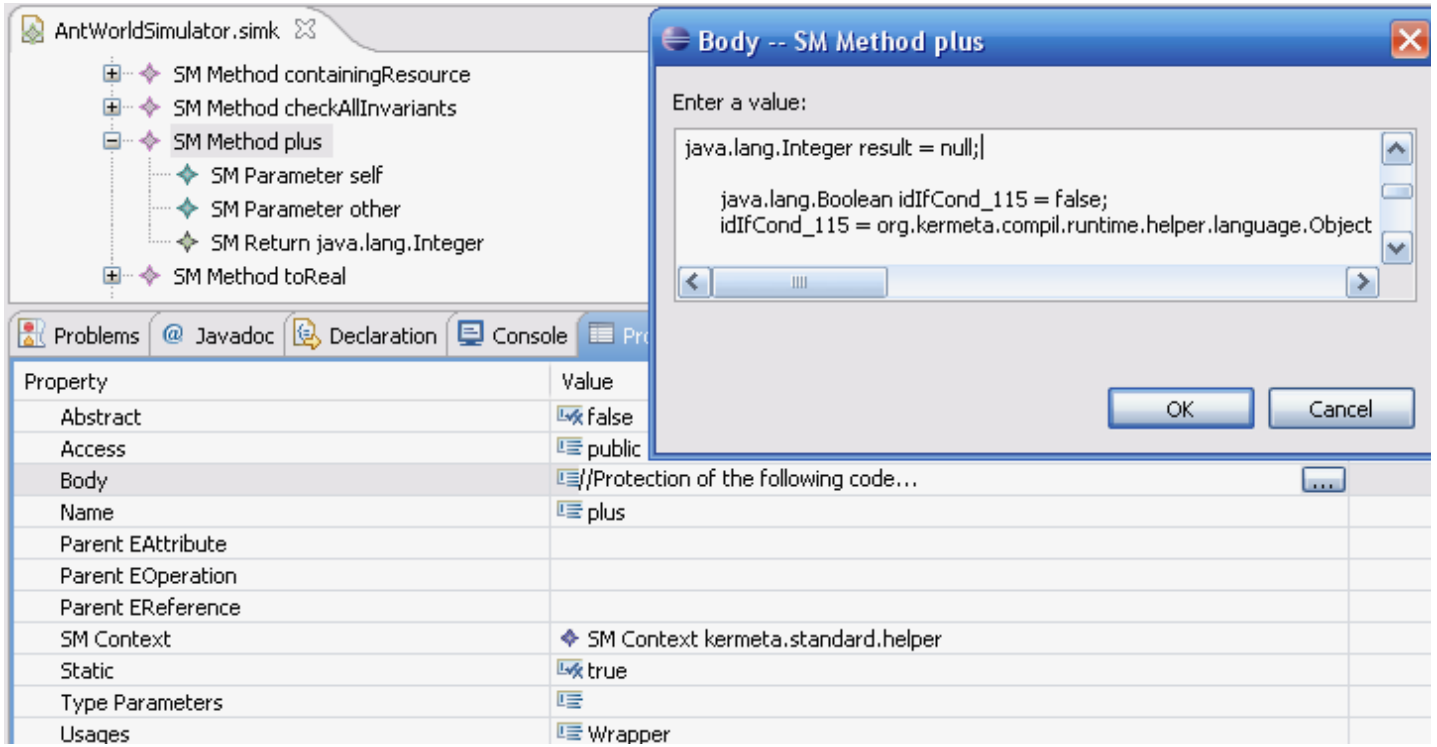
emf ECLIPSE MODELING FRAMEWORK

Need of a model in complement of Ecore

- The annotated Ecore model is not enough
  - e.g.: to handle the call of super operation in multi-inheritance context, we need static methods to call the given method
  - Static Methods are not supported by Ecore
- **Simk:** ***S****tatic* ***I****ndirection* ***M****odel for* ***K****ermeta*
  - developed for the compiler, but not dedicated
  - a new metamodel instead of file generation, why ?
    - to save at the end of the 2nd step the generated static methods
    - then use the generated sources at posteriori in the compilation process, i.e. the generation of the methods from Simk is performed after the EMF Java classes generation by using Jet templates

Need of a model to complement an *.ecore

- The Java Method and Java Class signatures are modeled, but the method body is a single String
- Simk model contains Java implementation for
  - runners to launch **runnable** operations
  - **multi-inheritance** support, **invariant**
  - **ValueType** wrappers, e.g. *plus()* from *Integer*

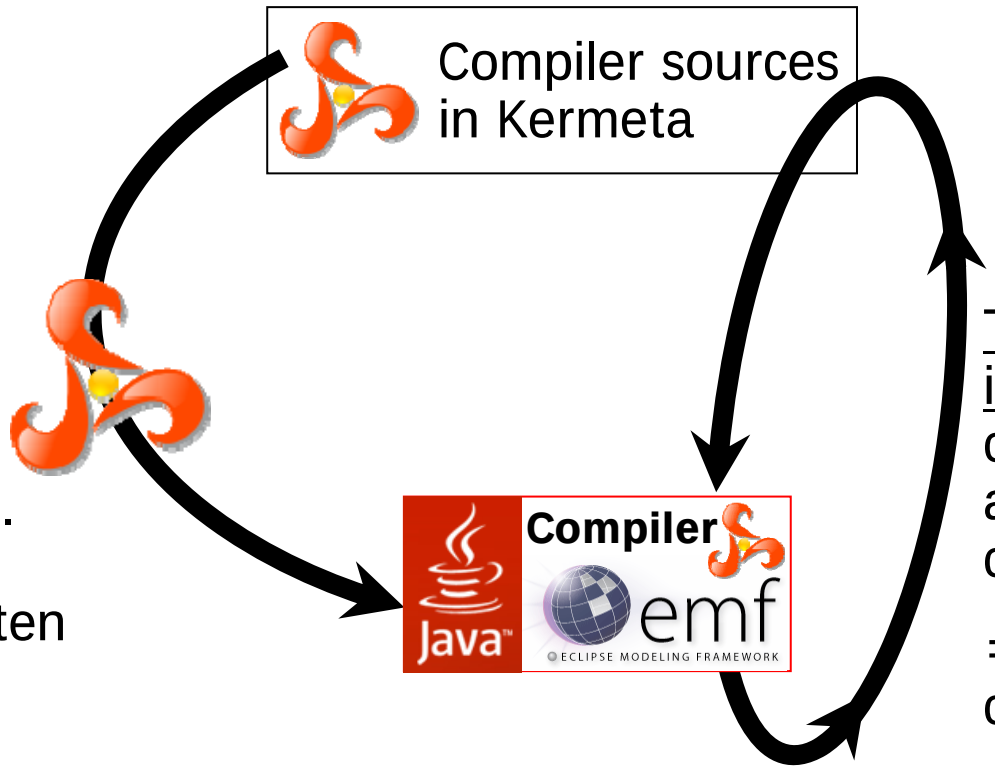


Implementation details

- The 2nd step of the compiler process (Km to Ecore + EAnnotations)
  - written in Kermeta as a model **transformation**
  - transformation in 2 passes
    - 1rst: creation of the Ecore elements
    - 2nd: creation of the links between the elements and operation behavior
  - Kermeta Aspect feature is used intensively
    - management of the traceability for keeping the source Km element corresponding to a new Ecore element
    - application of design patterns: visitor ...

Implementation details

- The compiler is **fully written** in Kermeta
- **Bootstrap**, the compiler compiles itself



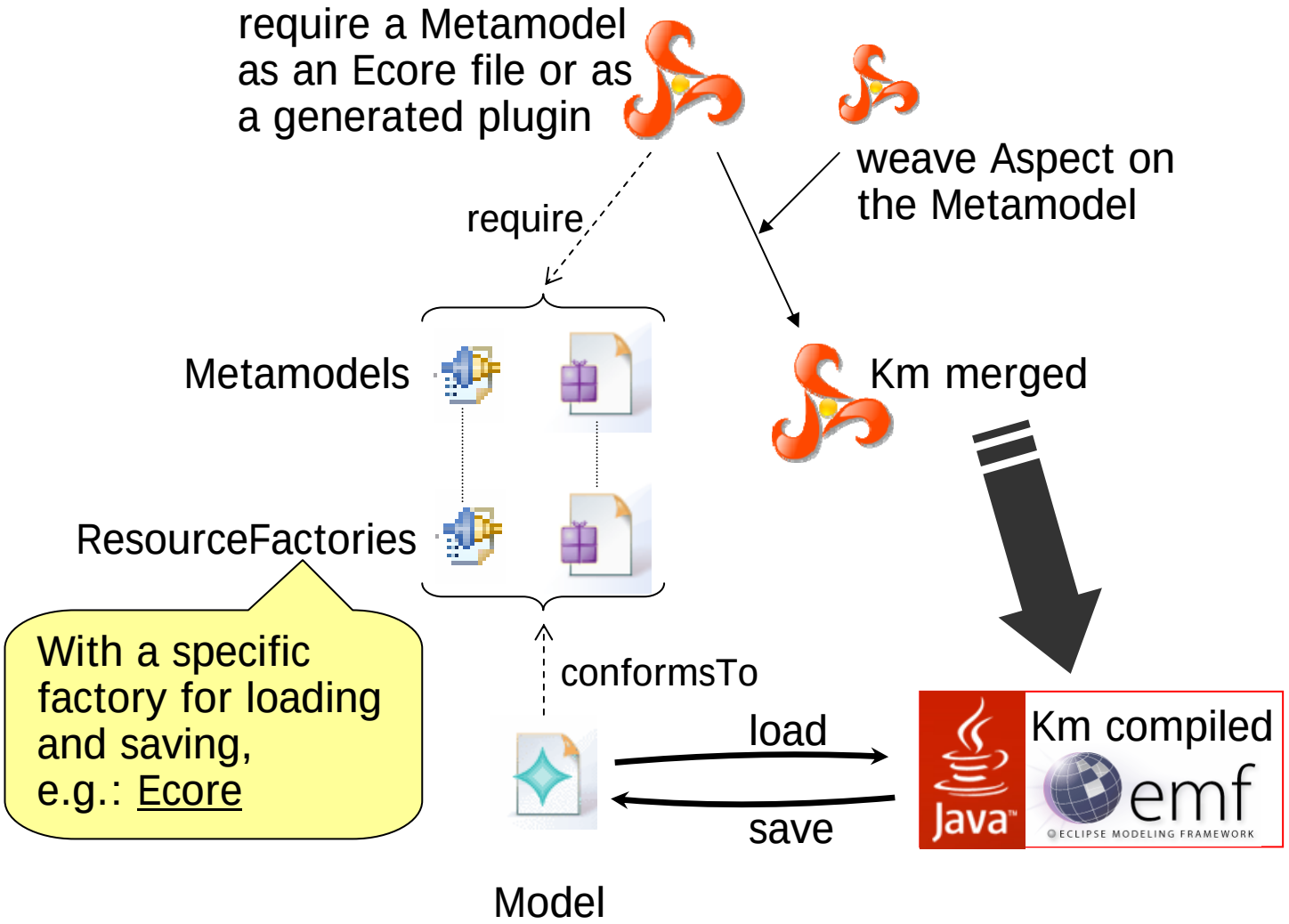
1rst iteration: the compiler sources are compiled in interpreted mode. Thus, it produces the compiler written in Java/EMF

The following iterations: the compiler sources are compiled in compiled mode

=> the compiler compiles itself

Implementation details

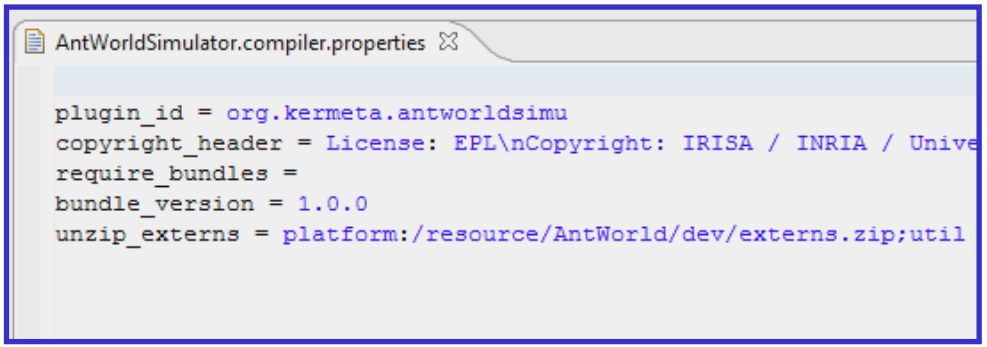
- Enabling the usage of a metamodel generated as a plugin or a simple Ecore metamodel for **persistence** issues



How to customize the compilation process

- Customizing the compilation process
  - used to automate recurrent settings and post-treatments
  - parameter values are contained in a *properties* file

- Settings for the genmodel
  - plugin_id
  - copyright_header



- Post-treatments
  - require_bundles (plugin dependencies)
  - bundle_version (plugin version)
  - main_operations [available in SVN version]
  - unzip_externs (including Java source codes given by the user) [available in SVN version]

Current limitations

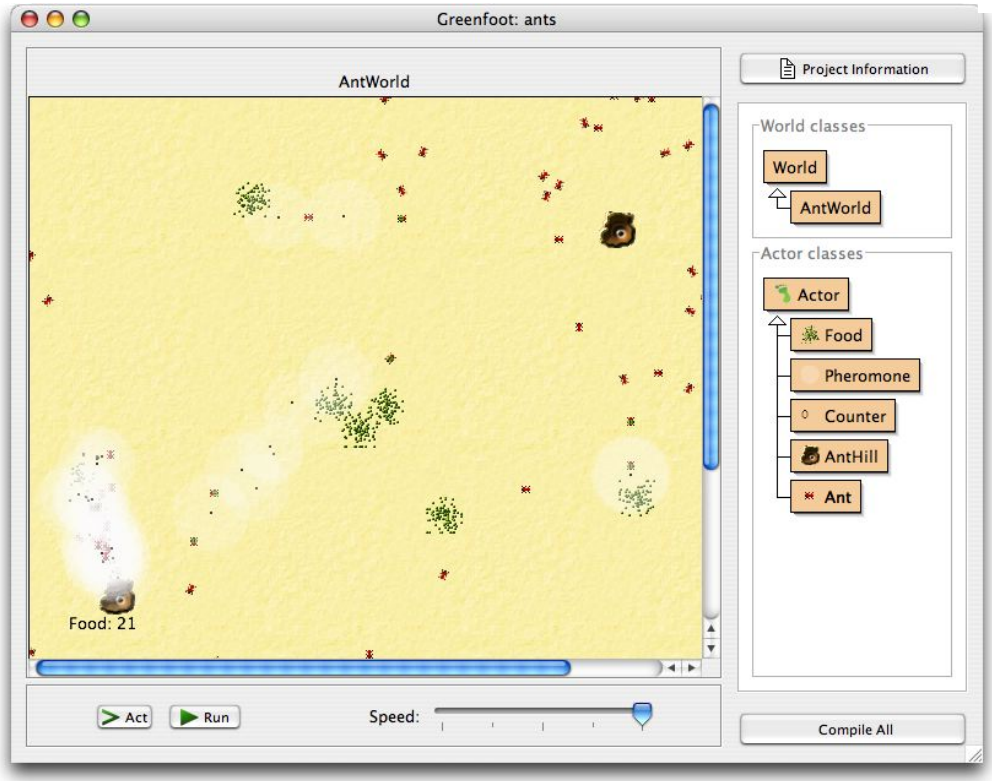
- Kermeta language features not supported
  - Model Typing
  - Dynamic Expression
  - Recursive function type
- The process is not incremental
  - the full process must be replayed for any Kermeta program modifications

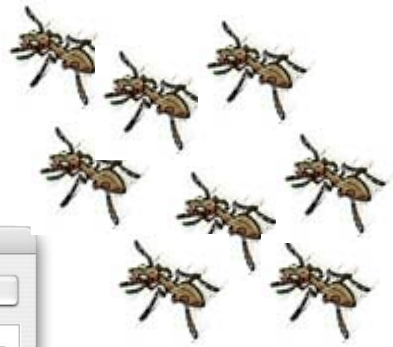
Experiments

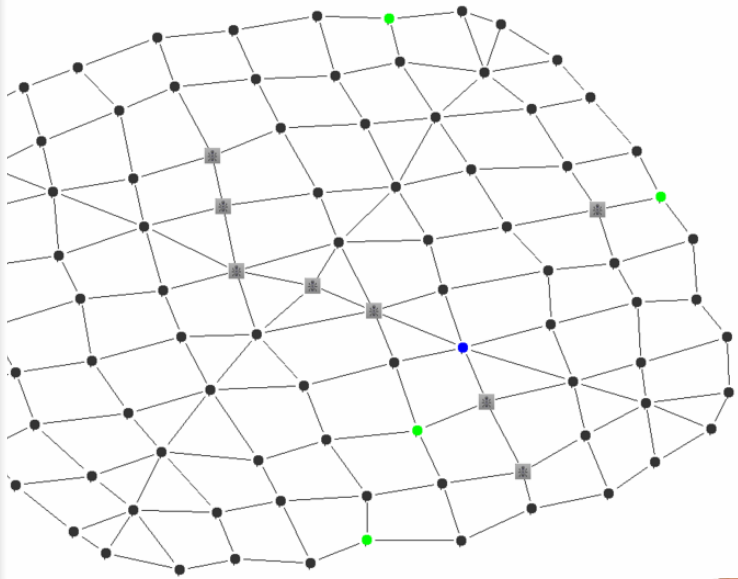
- AntWorld simulation (live demo)
- Kompose (reflexive algorithm)
- OCL to Kermeta transformation
- FSM (pre/post conditions and invariant)
- Ecore from XSD + XML files as input/output
- ...

Experiments: AntWorld Simulation

- Algorithm goal: evaluate tool performance
  - execution time
  - memory usage



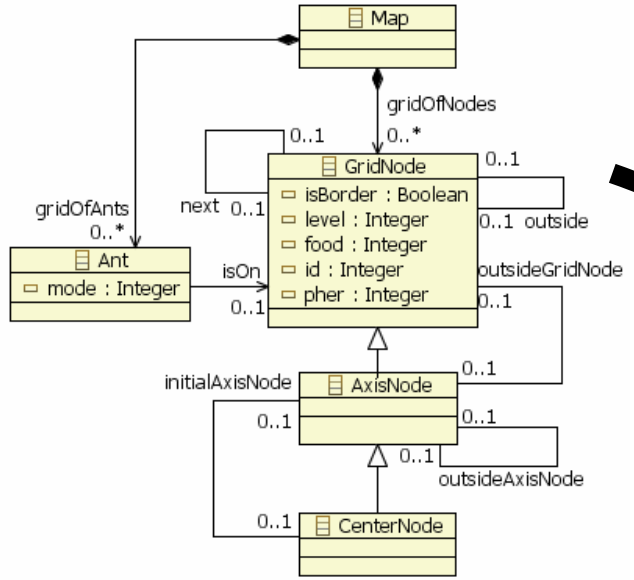




Source: GreenFoot

Experiments: AntWorld Simulation

- AntWorld simulation demo's content

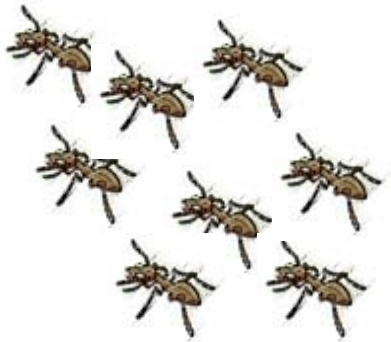


behavior added by Aspect

Generation of Ecore, Simk and Genmodel models (including specific properties)

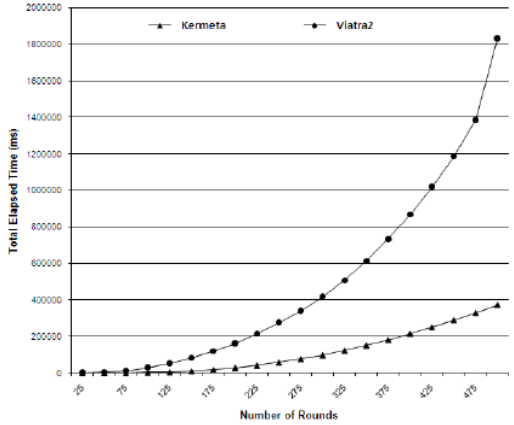
Execution as Java Application in Eclipse

Experiments: AntWorld Simulation



- Results
  - all the resources in a single plugin
  - **x50** faster than the interpreted mode

- Comparison with Graph Transformation tools
  - better results in terms of **execution time** than other tools based on Eclipse and eventually EMF like: **Viatra2 (x4,9)**, **EMF Transformation (x65)**
  - **best solution** in terms of memory usage: **VMTS (x3)**



Experiments

- AntWorld simulation (live demo)
- Kompose (reflexive algorithm)
- OCL to Kermeta transformation
- FSM (pre/post conditions and invariant)
- Ecore from XSD + XML files as input/output models
- ...

- Medium rate: **x35** faster than the interpreted mode

Conclusion

- Increase performance (x35 faster)
- Generate a Kermeta program as a Java/EMF plugin
- Improve the deployment process in industrial context
- Easy to use: a simple right-click

QUESTION ?

Try the compiler !

Download Kermeta 1.3.0

Documentation available on the Kermeta web site

http://kermeta.org/community/dev/compilerCompilingIssues