19th International Conference on Software and Systems Reuse, ICSR’20

Modular Moose

A new generation software reverse engineering environment

Nicolas Anquetil, A. Etien, M.H. Houekpetodji, B. Verhaeghe, S. Ducasse, C. Toullec, F. Djareddir, J. Sudich, M. Derras

Université de Lille

Inria informatics mathematics

RMod

Agenda

Reverse Engineering requirements

Famix: Generic Software Meta-Model

FamixNG: Composable Software Meta-Models

ModMoose: Integrated Reverse Engineering Environment

Conclusion

Agenda

Reverse Engineering requirements

Famix: Generic Software Meta-Model

FamixNG: Composable Software Meta-Models

ModMoose: Integrated Reverse Engineering Environment

Conclusion

Reverse Engineering

Many different tasks (navigation, visualization, querying, data/control flow, ...)

Huge data (millions of entities: classes, methods, variables, ...)

Highly specialized tools (costly to develop)

Reverse Engineering

- How can we reuse such tools?
- Across
  - Systems
  - Programming languages
  - Organizations
  - ...

Reverse Engineering

How can we reuse such tools?

Across

- Systems
- Programming languages
- Organizations
- ...

➡ **Generic Software Meta-Model**

Moose, some background

A platform for software analysis

Based on a generic meta-model (Famix)

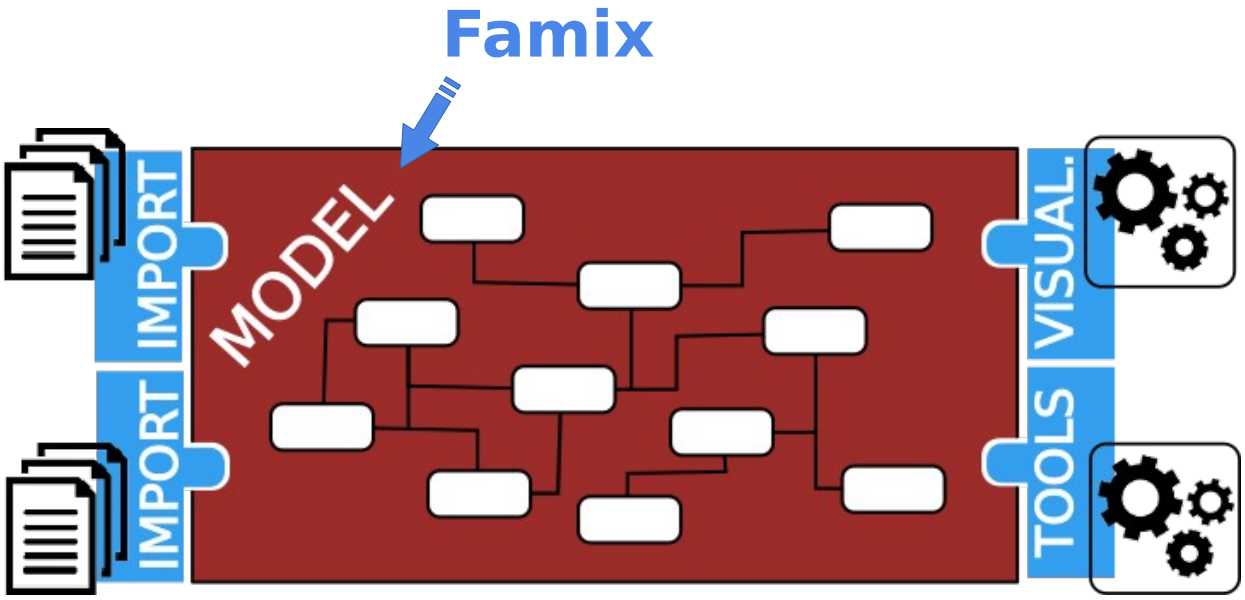
Developed since 1996

https://github.com/moosetechnology/moose-wiki

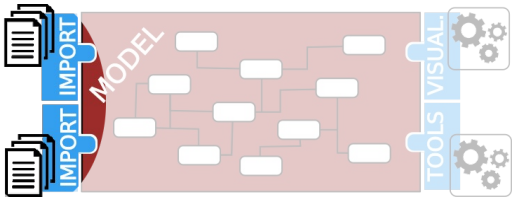
https://github.com/moosetechnology

Moose, some background

A platform for software analysis



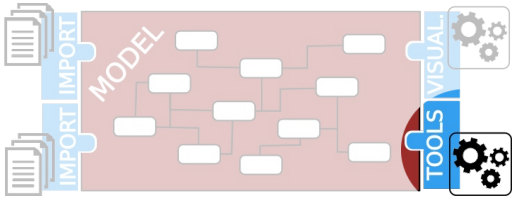
Moose, some background



Importers

- Source code (4D, Ada, C/C++, COBOL, Fortran, Java, Mantis, Pharo, PowerBuilder, SQL, TypeScript, VBA, ...)
- HTML, XML, CSV, ...
- Bugs, authors, commits

Moose, some background

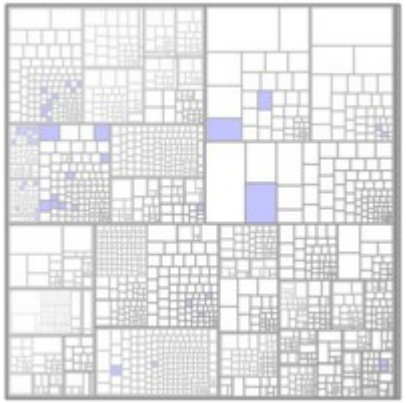


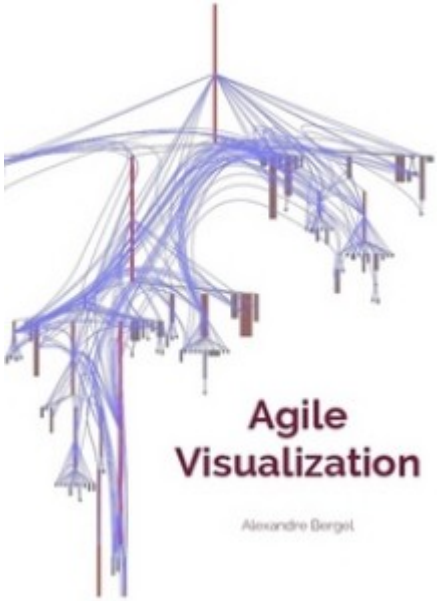
Tools

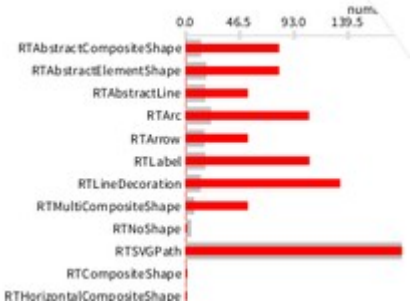
- MooseQuery: DSL to query/navigate a model
- Software engineering metrics: cohesion/coupling, Chidambler & Kemerer metrics, cyclomatic complexity, ...
- Data Mining algorithms

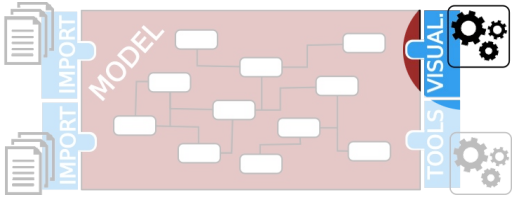
Moose, some background

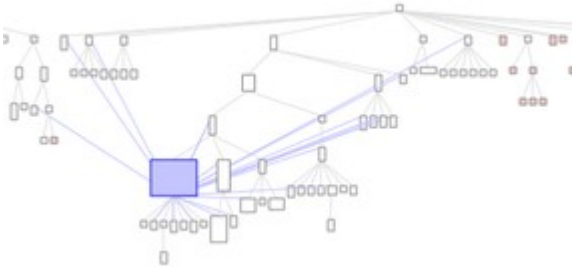
Visualizations











Agenda

Reverse Engineering requirements

Famix: Generic Software Meta-Model

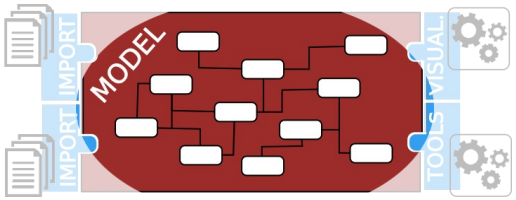
FamixNG: Composable Software Meta-Models

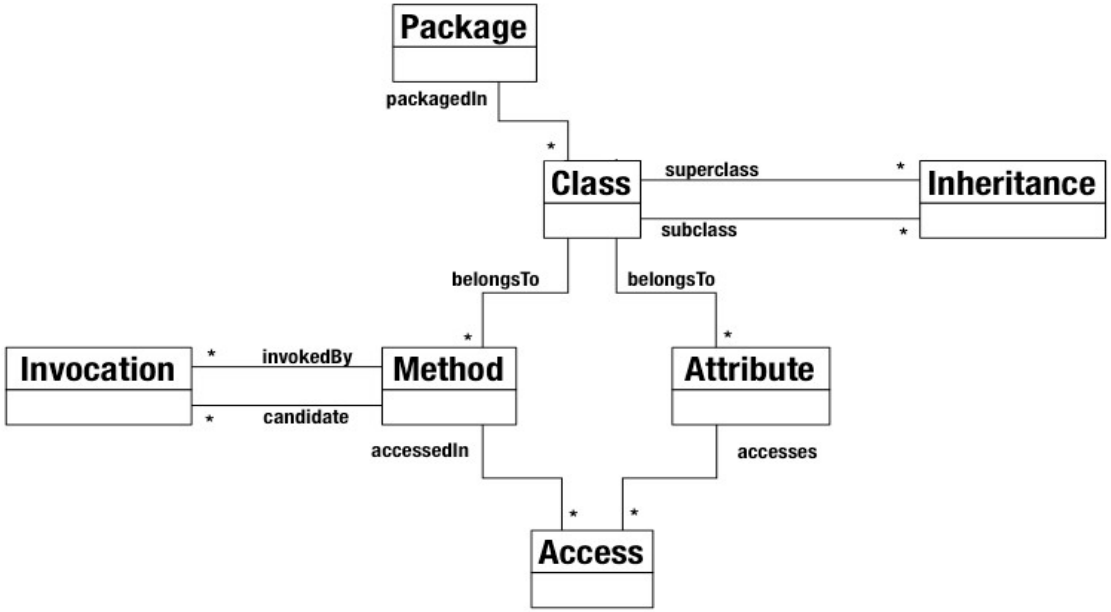
ModMoose: Integrated Reverse Engineering Environment

Conclusion

Software Meta-Modeling

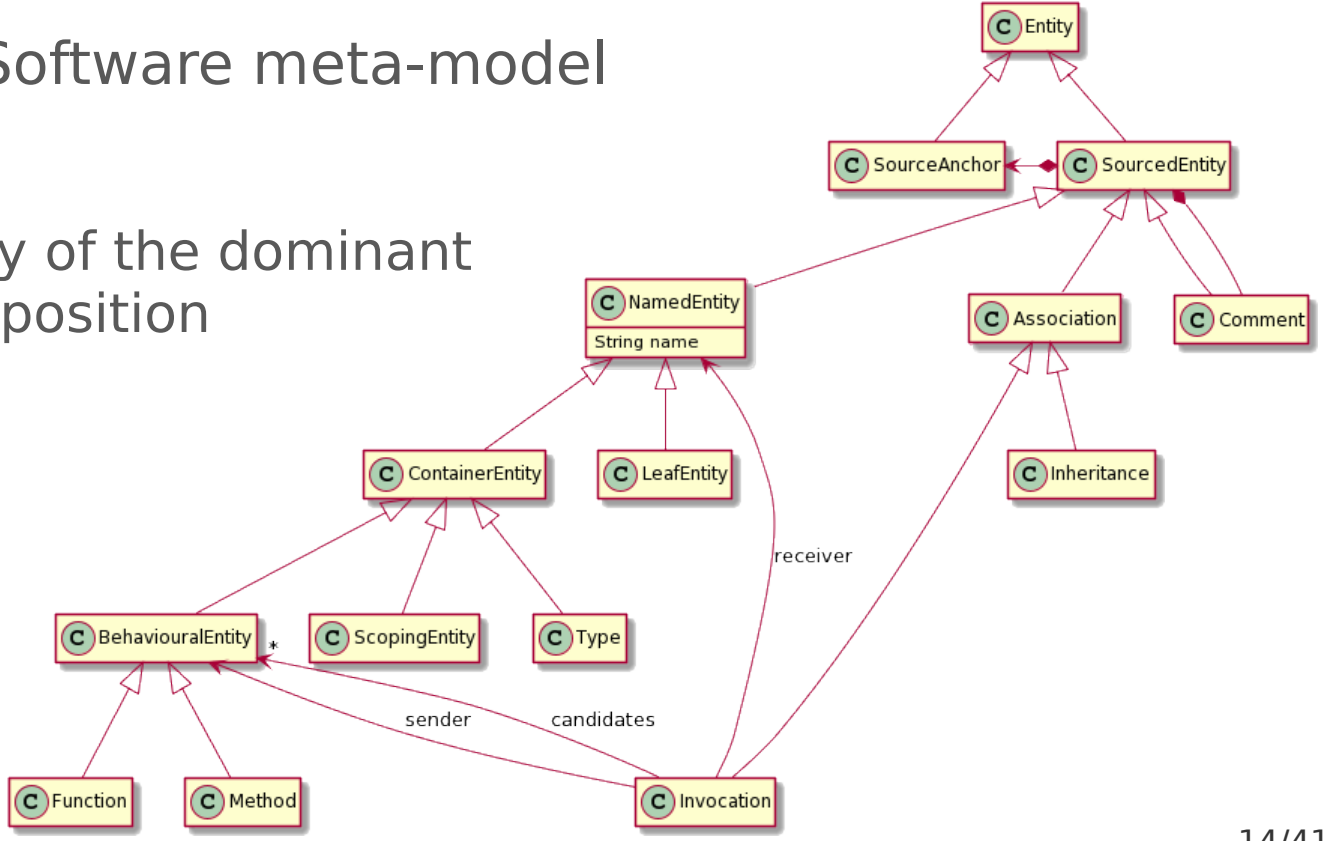
Generic software meta-model





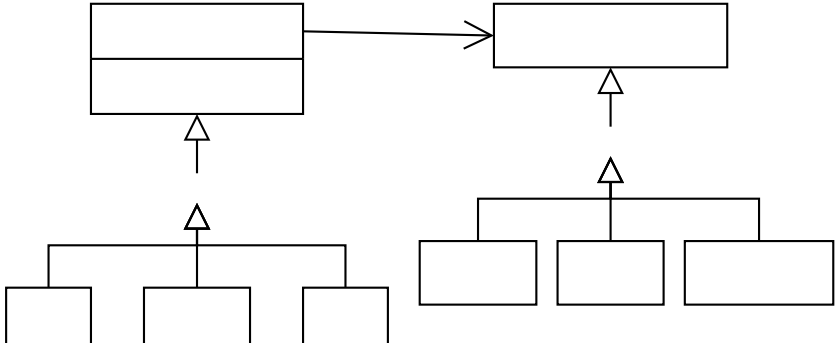
Software Meta-Modeling

- Generic Software meta-model
  - Famix
  - Tyranny of the dominant decomposition



Software Meta-Modeling

- Generic software meta-model
  - Dagstuhl Middle Meta-model
  - Tyranny of the dominant decomposition



Software Meta-Modeling

- Famix: Generic software meta-model ?
  - Entities have unneeded/misleading properties (package can be invocation receivers)
  - Difficult to evolve (add new language)
  - New languages could change existing entities

Agenda

Reverse Engineering requirements

Famix: Generic Software Meta-Model

FamixNG: Composable Software Meta-Models

ModMoose: Integrated Reverse Engineering Environment

Conclusion

Composable Meta-Model

~~Famix: One generic meta-model for all languages~~

Create specialized meta-model for each language

FamixNG: Bare bone entities + composable properties

Ex: Classes have *attributes* and *methods* and *inheritance*

+ *visibility* (public, private, protected, friend, ...)

+ *partial* classes + *extension* methods + *use* traits

Composable Meta-Model

- FamixNG
  - ≃ 100 traits (*i.e.* a set of methods that classes can *use* with a kind of “multiple inheritance”)

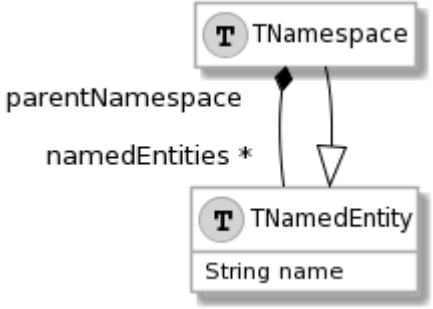
Composable Meta-Model

FamixNG

≃ 100 traits (*i.e.* a set of methods that classes can *use* with a kind of “multiple inheritance”)

Ex: `TNamedEntity` used by entities that have a name

Ex: `TNamespace` used by entities that contain `TNamedEntity`



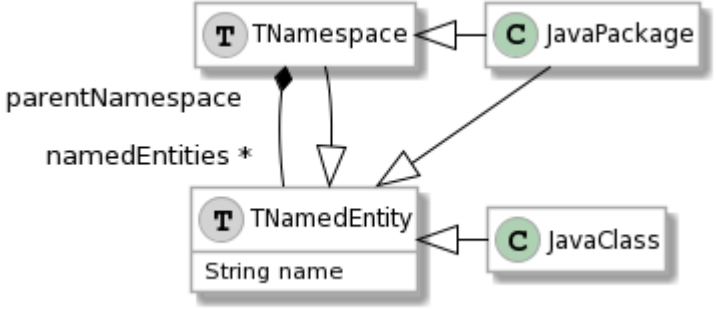
Composable Meta-Model

FamixNG

≃ 100 traits (*i.e.* a set of methods that classes can *use* with a kind of “multiple inheritance”)

Ex: `TNamedEntity` used by entities that have a name

Ex: `TNamespace` used by entities that contain `TNamedEntity`



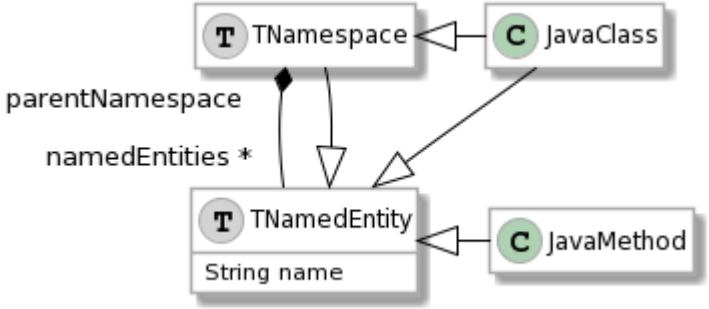
Composable Meta-Model

FamixNG

≃ 100 traits (*i.e.* a set of methods that classes can *use* with a kind of “multiple inheritance”)

Ex: `TNamedEntity` used by entities that have a name

Ex: `TNamespace` used by entities that contain `TNamedEntity`



Composable Meta-Model

Generic `TClass` in FamixNG *uses*:

- `TType` (which *uses* `TNamedEntity`)
- `TWithComment`
- `TPackageable`
- `TWithInheritances`
- `TWithAttributes`
- `TWithMethods`
- `TInvocationsReceiver`

Composable Meta-Model

`JavaClass` *uses*:

- `TClass`
- `TWithVisibility`
- `TWithExceptions`

`PharoClass` *uses*:

- `TClass`
- `TUsesTraits`
- `TWithExtentions`

Composable Meta-Model

FamixNG traits library

Relation traits (7):
Inheritance, Access, Invocation, Reference, UseTrait, IncludeFile

Technical traits (12):
SourceAnchor, metrics, queries

Property traits(46):
name, comments, typed, visibility, invocable, ...

Entity traits (38):
Class, Method, Parameter, Exception, Function, ...

Meta-Model builder

New language meta-models built from FamixNG traits library

“Builders” to create metamodels

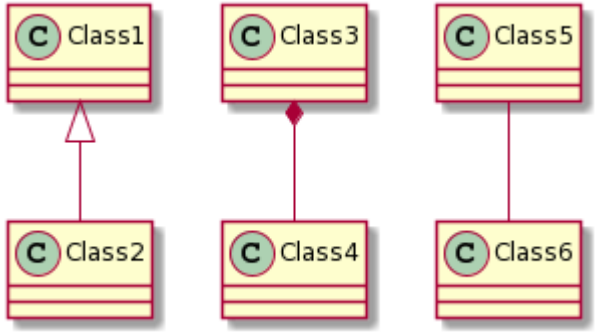
Meta-Model builder

New language meta-models built from FamixNG traits library

“Builders” to create metamodels

DSL (inspired by PlantUML.com)

```
@startuml
Class1 <|-- Class2
Class3 *-- Class4
Class5 — Class6
@enduml
```



Meta-Model builder

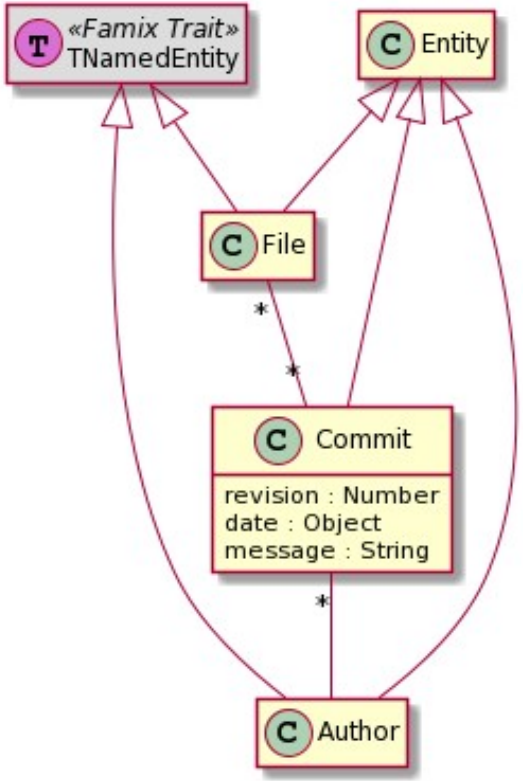
Example: Meta-model for commits

Define classes

```
entity := builder newClassNamed: 'Entity'.
file := builder newClassNamed: 'File'.
commit := builder newClassNamed: 'Commit'.
author := builder newClassNamed: 'Author'.
```

Define properties

```
commit property: 'revision' type: #Number.
commit property: 'date' type: #Object.
commit property: 'message' type: #String.
```



Meta-Model builder

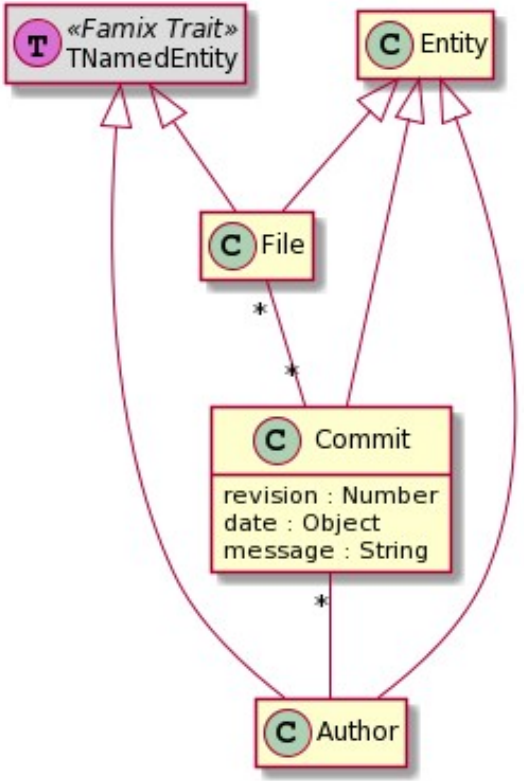
Example: Meta-model for commits

Define inheritances

```
file --|> entity.
file --|> TNamedEntity.
commit --|> entity.
author --|> entity.
author --|> TNamedEntity.
```

Define relations

```
file *-* commit.
commit *- author.
```



Agenda

Reverse Engineering requirements

Famix: Generic Software Meta-Model

FamixNG: Composable Software Meta-Models

ModMoose: Integrated Rev. Engin. Environ.

Conclusion

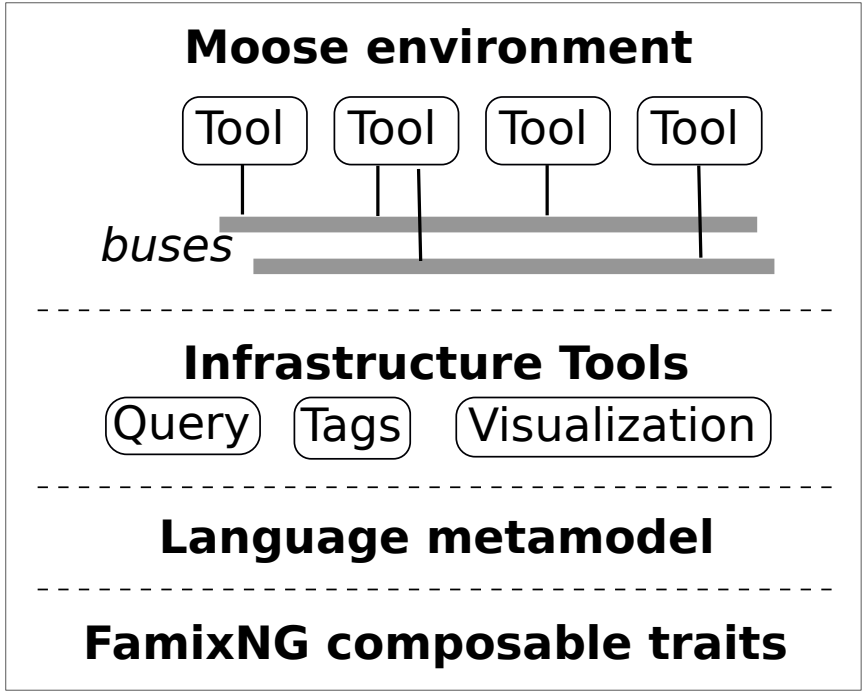
Integrated Reverse Engineering Environment

- Reverse engineering involves many tasks
  - Visualization, query, metrics, navigation, dependency analysis, control flow/data flow analysis

- Many specialized tools
- Generic tools (across meta-models)
- Adaptable, collaborating, tools

ModMoose

Architecture of an Integrated Reverse Engineering Environment



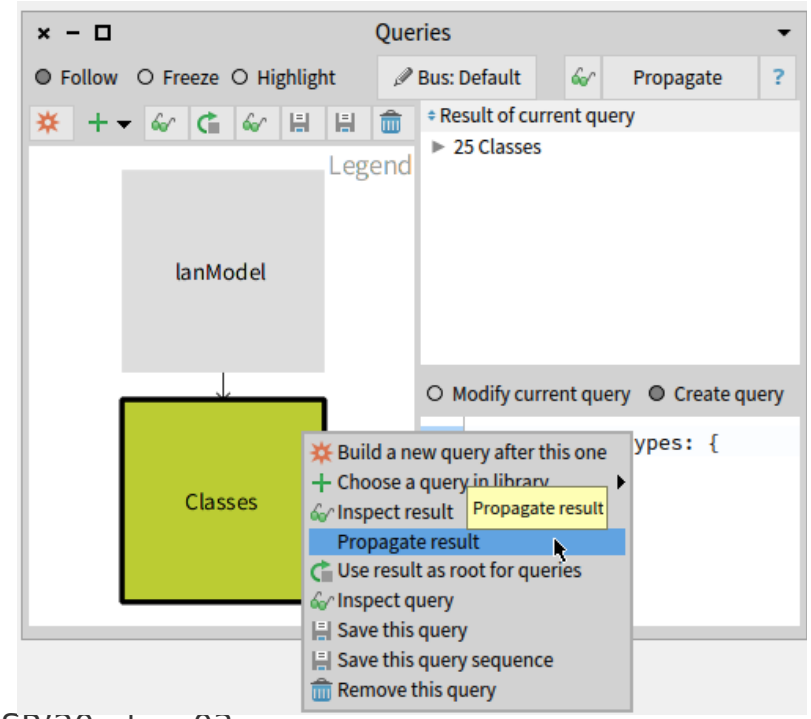
ModMoose

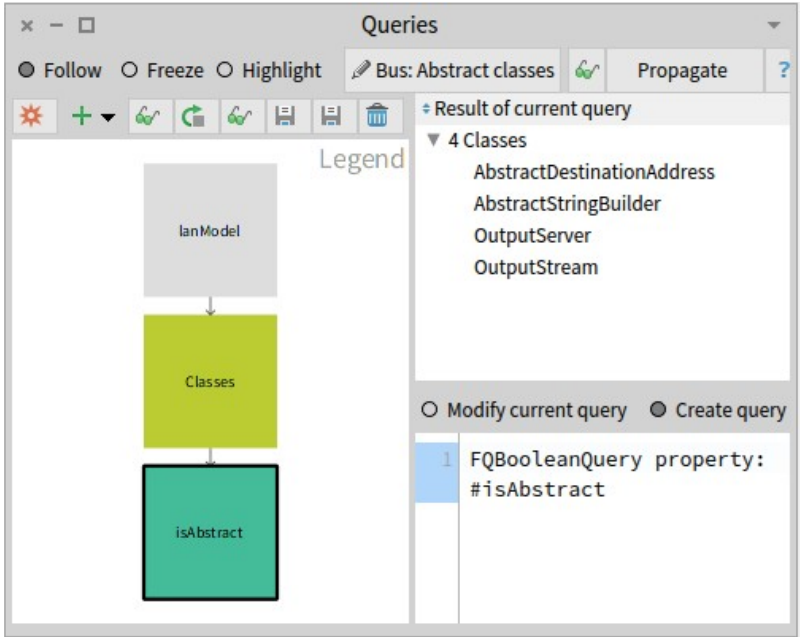
ModMoose rules:

- The environment centralizes data and tool interactions
- Tools are focusing on a single task: *e.g.*, the Query Browser works on a set of model entities and produces another set of entities
- Tools communicate through buses, they “read” model entities on their bus(es) and “write” entities back on their bus(es)

Tools

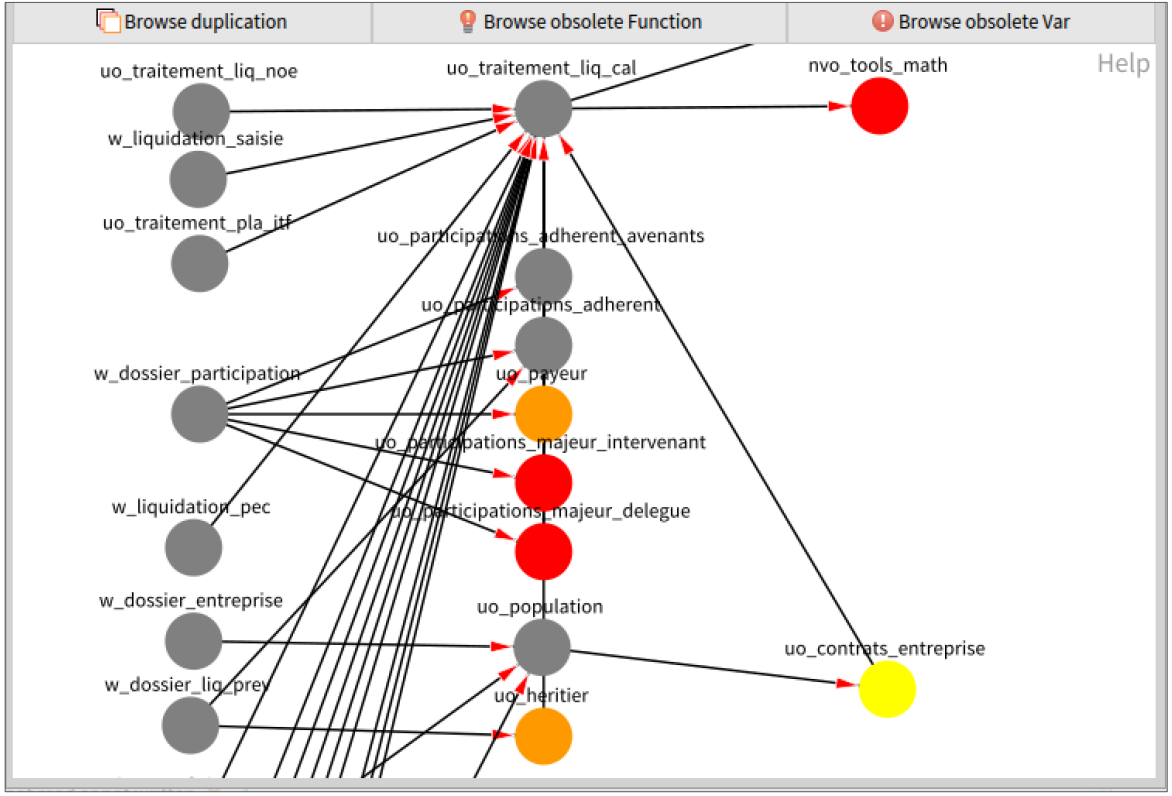
Query Browser





Tools

Dependency Graph Browser



Tools

Entity Inspector

Navigation

- #'Smalltalk::AbstractFont'
- #'Smalltalk::AnnouncementsAPIDocumentation'
- #'Smalltalk::AnnouncementsHelp'
- #'Smalltalk::AnnouncementsTutorial'
- #'Smalltalk::Array'
- #'Smalltalk::ArrayedCollection'
- #'Smalltalk::AthensAbstractPaint'
- #'Smalltalk::AthensAbstractShape'
- #'Smalltalk::AthensAffineTransform'
- #'Smalltalk::AthensBalloonEngine'
- #'Smalltalk::AthensBalloonGradientPaint'
- #'Smalltalk::AthensBalloonImagePaint'
- #'Smalltalk::AthensBalloonLinearGradient'
- #'Smalltalk::AthensBalloonPaint'
- #'Smalltalk::AthensBalloonPath'
- #'Smalltalk::AthensBalloonRadialGradient'
- #'Smalltalk::AthensBalloonSolidColorPaint'
- #'Smalltalk::AthensBalloonStrokePaint'
- #'Smalltalk::AthensBalloonSurface'
- #'Smalltalk::AthensCairoCanvas'

Navigation | Tree | Moose Properties | Fame

Name	Type	Opposite	Derived?	Container?	IsTarget?	IsSource?
annotationInstan	TAnnotationInsta	annotatedEntity	true	false	false	false
attributes	TAttribute	parentType	true	false	false	false
comments	TComment	container	true	false	false	false
declaredSourceL	TSourceLanguag	sourcedEntities	false	false	false	false
definedAnnotatic	TAnnotationType	annotationTypes	true	false	false	false
exceptions	TException	exceptionClass	true	false	false	false
extendedMethod	Method		true	false	false	false
incomingReferen	TReference	target	true	false	false	false
methods	TMethod	parentType	true	false	false	false
parentPackage	TPackage	childEntities	false	true	false	false
receivingInvocati	TInvocation	receiver	true	false	false	false
sourceAnchor	TSourceAnchor	element	true	false	false	false
subInheritances	TInheritance	superclass	true	false	false	false
superInheritance	TInheritance	subclass	true	false	false	false
typeContainer	TWithTypes	types	false	true	false	false
typedEntities	TTypedEntity	declaredType	true	false	false	false
types	TType	typeContainer	true	false	false	false

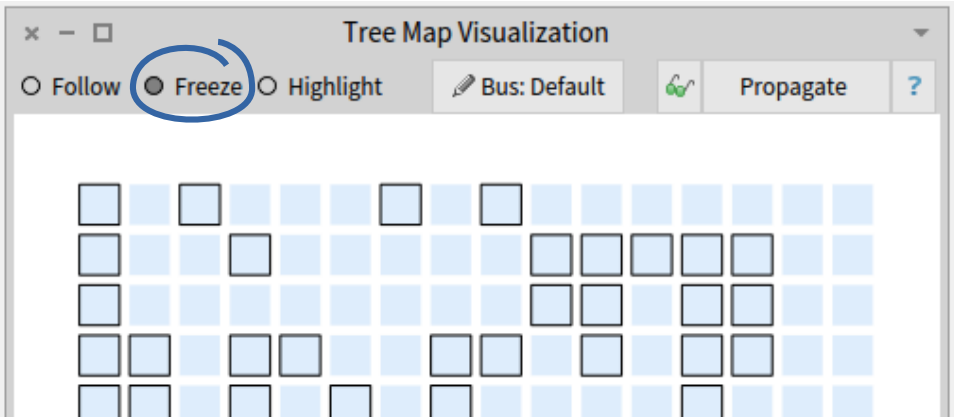
Communication Buses

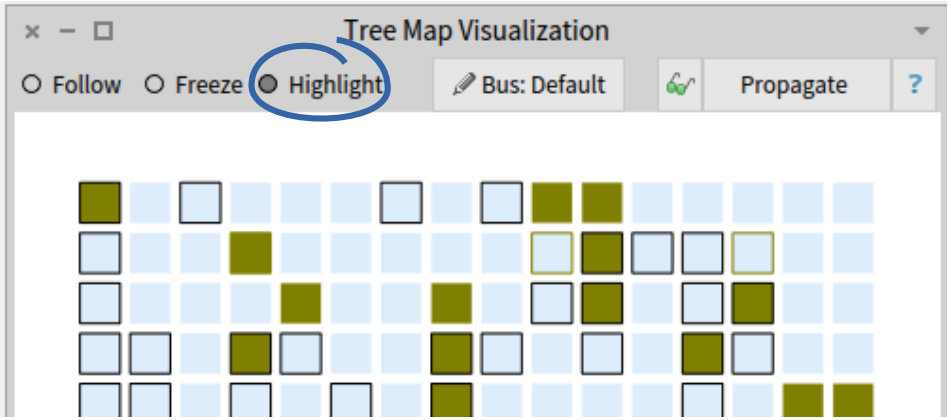
- Tools “read” and “write” model entities on buses
- Each tool can be attached to 0 to n buses
- Can have several instances of the same tool on different buses
- Ex: Compare dependencies of two set of entities
  - Bus1: QueryBrowser1 + DependencyGraphBrowser1
  - Bus2: QueryBrowser2 + DependencyGraphBrowser2

Tools

Fine control of tool behavior

- Follow: Display incoming entities, produces outgoing entities
- Highlight: Highlight incoming entities in “frozen” display
- Frozen: Frozen display but, produces outgoing entities





Agenda

Reverse Engineering requirements

Famix: Generic Software Meta-Modeling

FamixNG: Composable Meta-Model

ModMoose: Integrated Reverse Engineering Environment

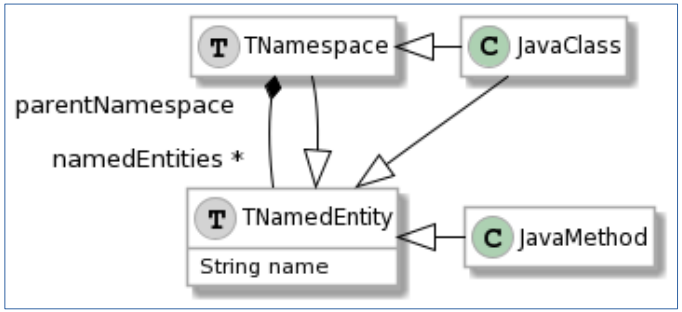
Conclusion

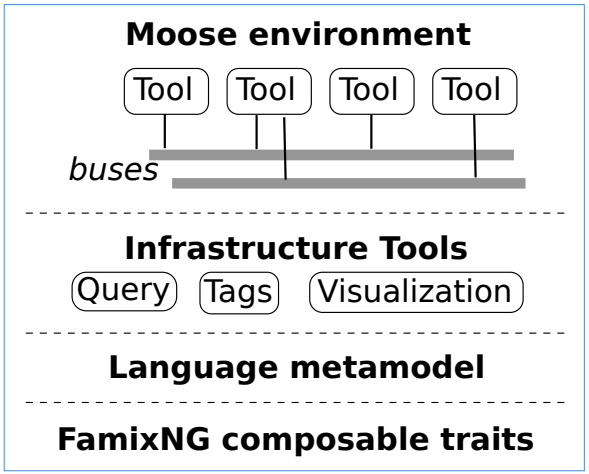
WrapUp

- FamixNG
  - Composition of programming language meta-models from basic traits
- ModMoose
  - Interactive Reverse Engineering Environment
  - Specialized tools communicating through buses

Modular Moose

Nicolas Anquetil
nicolas.anquetil@inria.fr





Université de Lille

Inria informatics mathematics

RMod