Moose
A software analysis tool

Current issues

- Issue 1: Scaling
  - 20mn are currently needed to load a 1 million elements model.
    - Such a model corresponds to industrial model with 6,000 classes and 75,000 methods.
  - Details of the methods are not in the model
    - But needed for example for language migration
- Issue 2: Initially thought for OO languages
  - Nowadays, lot of other languages (ADA, C, SQL…)
  - Tools and services are dedicated to OO (MooseChef, FAMIXDiff, Orion…)
  - Need to be easily extended
- Issue 3: Inadapted user syntax
  - Creating a new metamodel is tedious and error prone
  - Entrance ticket for using, learning and teachning Moose is very expensive

Moose at a glance

+	Generic metamodel	~	Modularity
+	Lot of tools	~	Memory print
+	Lot of developers	-	Compatibility with standarts
+	User community	-	Dedicated syntax

Issues in details – Creating a concept

- Pragma issue

```
FAMIXSQLFunction class >> annotation
    <MSEClass: #SQLFunction super: #FAMIXBehaviouralEntity>
      <package: #FAMIX>
      ^self
```

FAMIX prefix is needed but not everywhere

Issues in details – Creating a relation

- Pragma issue

  `<MSEProperty: #name type: #type opposite: #opposite> <multivalued> <derived>`

  - `<multivalued>` specifies the upper bound multiplicity however nothing for the lower bound multiplicity or specific value
    - Useful to specify something mandatory
    - Useful for automatic generation

  - `<derived>` When is it needed?
    - Can lead to problems while writing or loading an MSE.

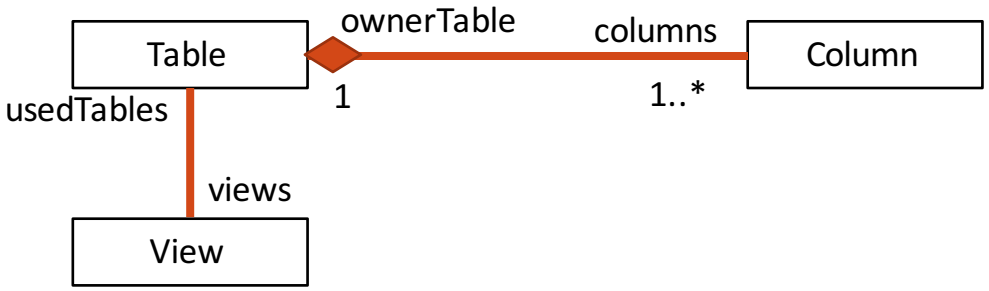
Issues in details – Creating a relation

- Updating the collections using FMMultiValueLink

```
ownerTable: aFAMIXTableOrNil
      ownerTable := FMMultivalueLink
            on: self
            update: #columns
            from: self ownerTable
            to: aFAMIXTableOrNil
```

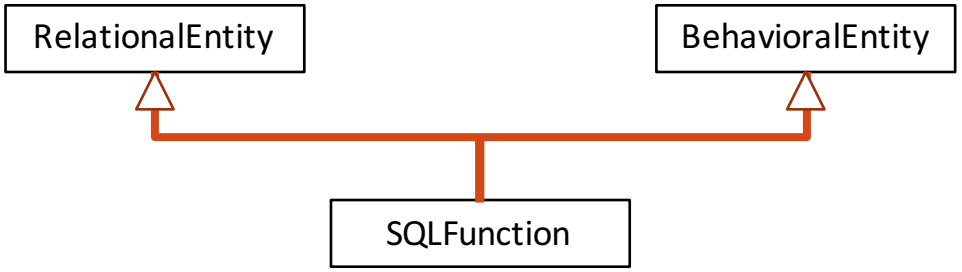
  - Where is the FMMultivalueLink needed?
  - What to do in case of n-m relationship?

Issues in details – Relationship vs ownership



- Currently no way to distinguish *normal* relationship and ownership
- `belongsTo` is often used but…
  - no specificity in the metamodel (before MooseQuery)
  - Methods in Pharo can belong to Class or Package.
- Useful to query at different abstraction level for instance

Issue in details – Multiple inheritance



- Omnipresent in metamodels
- Not supported as such in Pharo

What we want

Moose 2.0 = Moose 1.0
+ new meta and meta-metamodel
+ better memory print
+ simpler user syntax

What we want in details

Moose 2.0 = Moose 1.0
+ flexible metamodel (M)
  supporting multiple inheritance and type union (PL)
+ lot of visualization and analysis tools (M)
+ tool infrastructure (M)
+ model generator (PL)
+ compatibility with industrial and academic standarts (PL)
+ small memory print (P)
+ simpler user syntax (PL)
+ on demand (P) access to method details (M)