FAME

Meta-Modeling at Runtime

Adrian Kuhn

How to Model an Eternal Beer Store?

An eternal system can be extended at runtime.

An eternal application grows with your business.


Heineken
IMPORTED
THE ORIGINAL QUALITY
HEINEKEN LAGER BEER
TRADE MARK
Heineken
PREMIUM QUALITY

instance of ▸

Heineken	
name	= 'Heineken'
alc_vol	= 5
price	= 0.60
size	= 0.33

instance of

```
Beer
properties = {
  name     : String,
  alc_vol  : Number,
  price    : Number,
  size     : Number }
```

inst

```
Heineken : Beer
name     = 'Heineken'
alc_vol  = 5
price    = 0.60
size     = 0.33
```

instance of

```
Fame-Class
properties = {
  properties:Property[] }
```

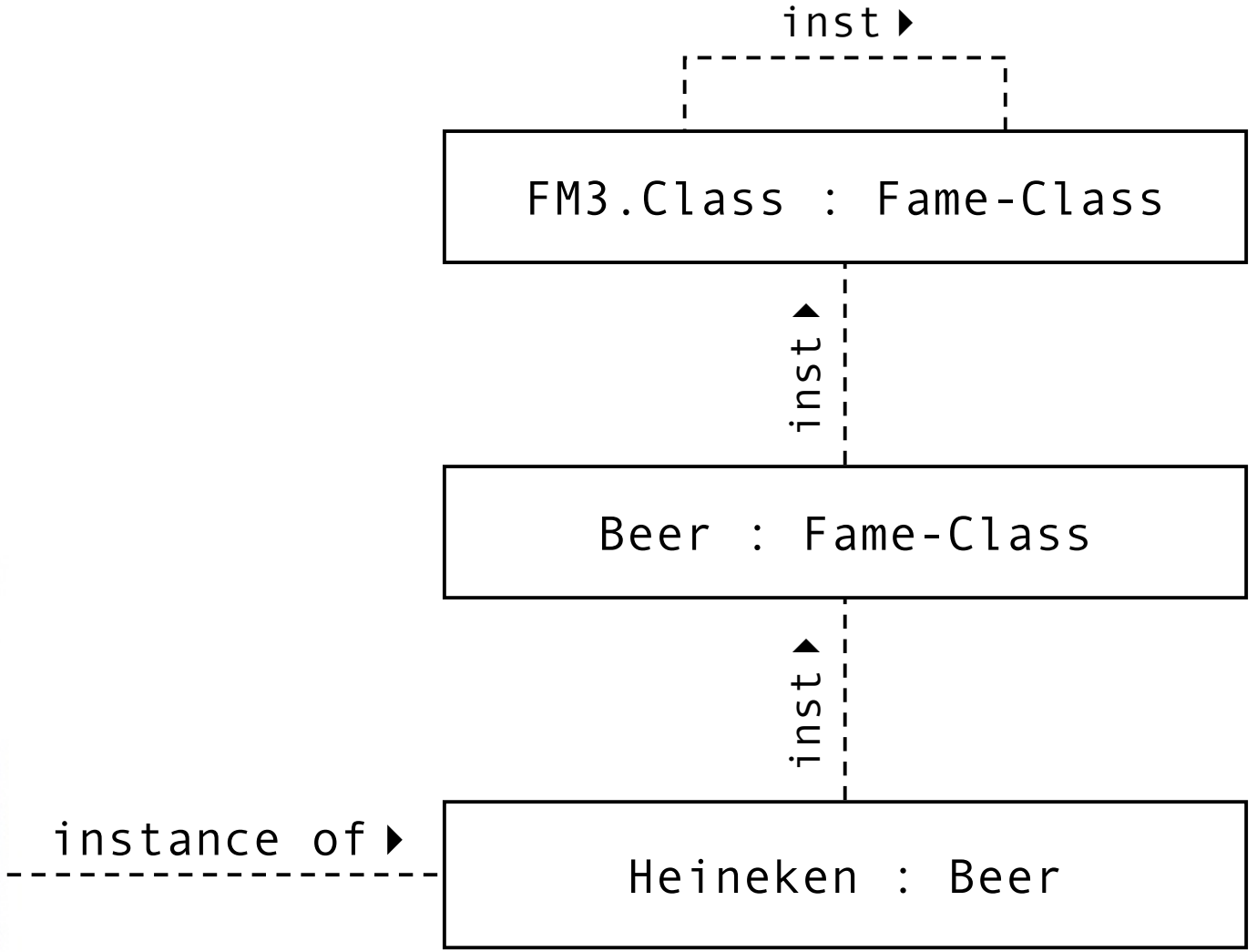
inst

```
Beer : Fame-Class
properties = {
  name:String, …, … }
```

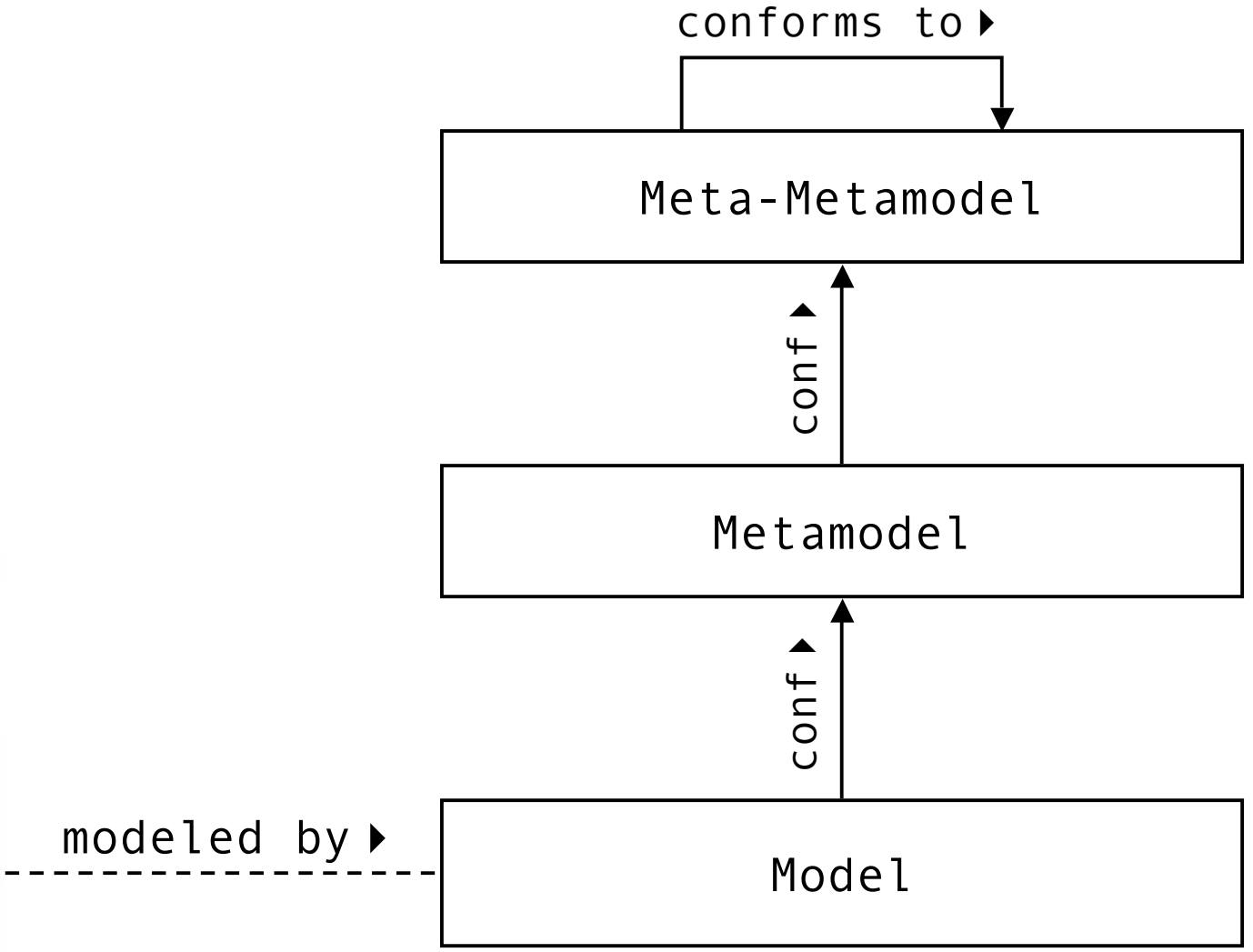
inst

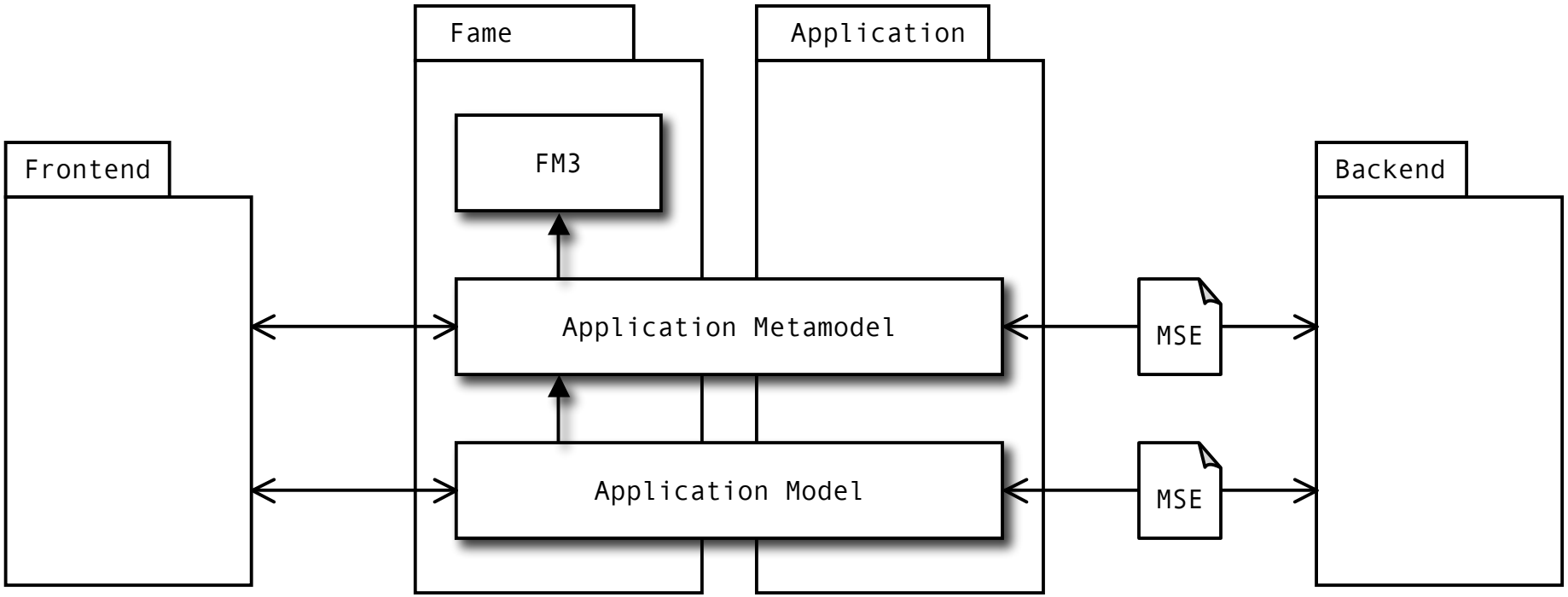
```
Heineken : Beer
name = 'Heineken'
…
```


Heineken
IMPORTED
THE ORIGINAL QUALITY
HEINEKEN LAGER BEER
Heineken
PREMIUM QUALITY


inst
FM3.Class : Fame-Class
inst
Beer : Fame-Class
inst
instance of
Heineken : Beer


Heineken
IMPORTED
THE ORIGINAL QUALITY
HEINEKEN LAGER BEER
Heineken
PREMIUM QUALITY


conforms to
Meta-Metamodel
conf
Metamodel
conf
modeled by
Model



Architecture of a typical Fame-based system.


```
heineken : HNK.Beer
```

```
name = 'Heineken'
alc_vol = 5
in_stock = 1000
price = 0.60
vol = 0.33
```

```
anOrder : HNK.Order
```

```
amount = 12
customer = joe
date = 2008-08-26 10:20
item = heineken
\total = 7.20
```

```
joe : HNK.Customer
```

```
name = 'Joe Example'
address = 'Levittown'
orders = { anOrder, … }
```


```
((HNK.Beer (id: 1)
    (name 'Heineken')
    (alc_vol 5)
    (in_stock 1000)
    (price 0.6)
    (vol 0.33))
(HNK.Order (id: 2)
    (amount 12)
    (customer (ref: 4))
    (date '2008-08-26 10:20')
    (item (ref: 1))
(HNK.Order (id: 3)
    (amount 6)
    (customer (ref: 4))
    (date '2008-08-23 13:45')
    (item (ref: 1))
(HNK.Customer (id: 3)
    (name = 'Joe Example')
    (address = 'Levittown')
    (orders (ref: 2)))
```

MSE File of Model

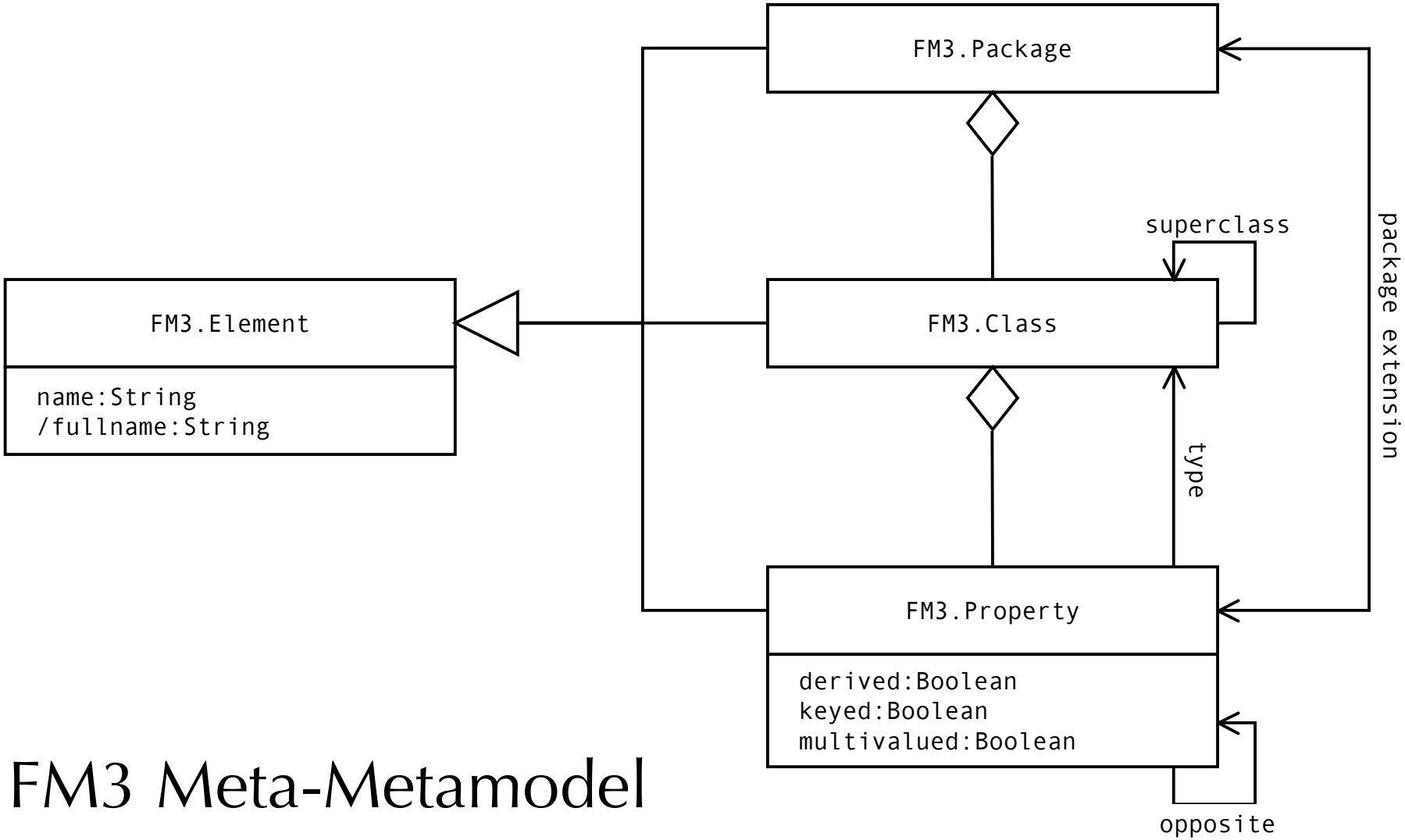
HNK.Beer
name:String alc_vol:Number in_stock:Number price:Number vol:Number

HNK.Order
amount:Number customer:Customer date:String item:Beer /total:Number

HNK.Customer
name:String address:String orders:Order[]

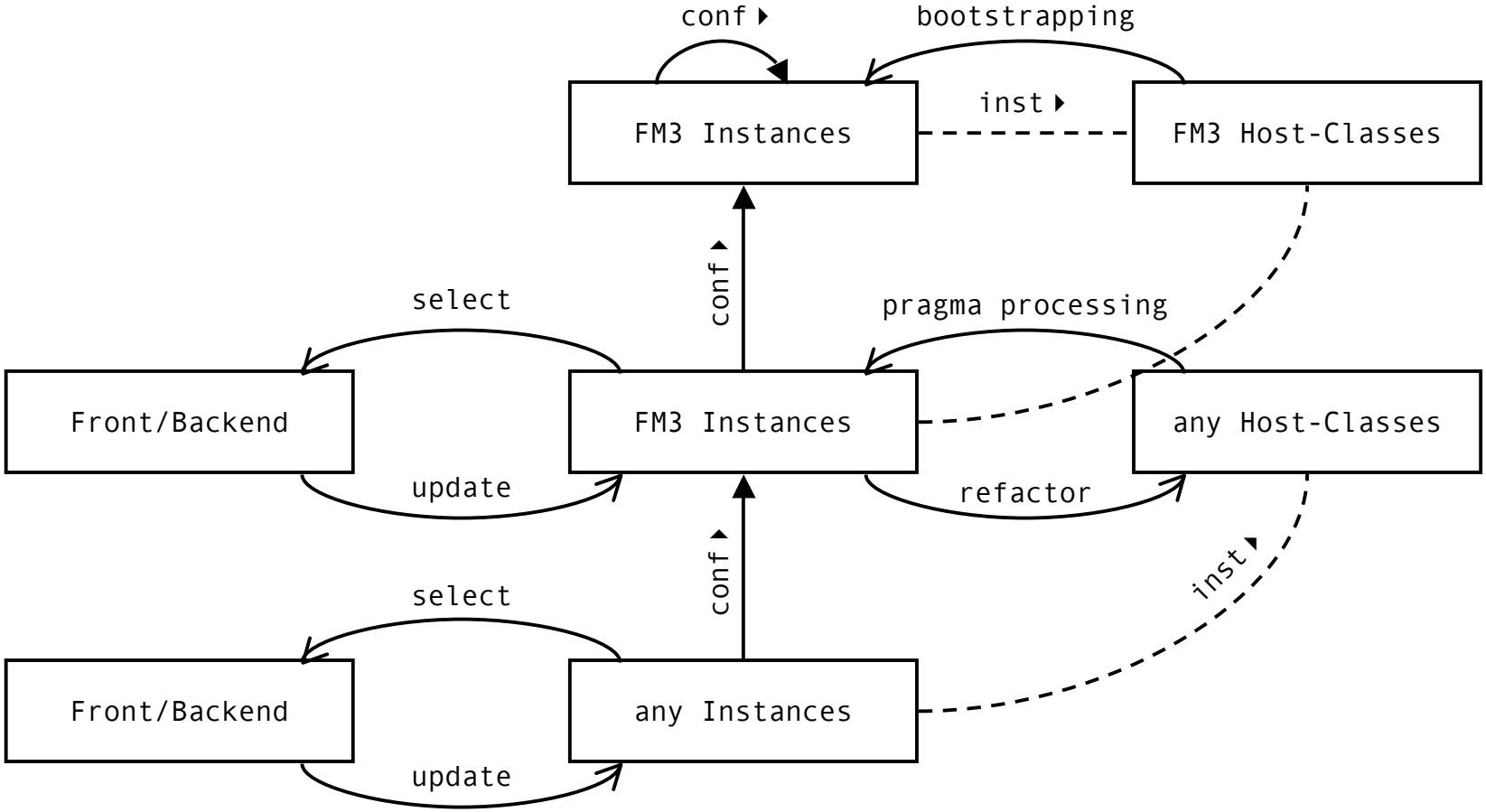

```
((FM3.Package (id: 1) (name 'HNK')
  (classes
    (FM3.Class (name 'Beer')
      (properties
        (FM3.Property (name 'name')
          (type (ref: 'String'))
        (FM3.Property (name 'alc_vol')
          (type (ref: 'Number'))
        (FM3.Property (name 'in_stock')
          (type (ref: 'Number'))
        (FM3.Property (name 'price')
          (type (ref: 'Number'))
        (FM3.Property (name 'vol')
          (type (ref: 'Number'))))
    (FM3.Class (id: 2) (name 'Order')
      (properties
        (FM3.Property (name 'amount')
          (type (ref: 'Number'))
        (FM3.Property (name 'customer)
          (type (ref: 3))
        (FM3.Property (name 'date')
          (type (ref: 'String'))
        (FM3.Property (name 'item')
```

MSE File of Metamodel

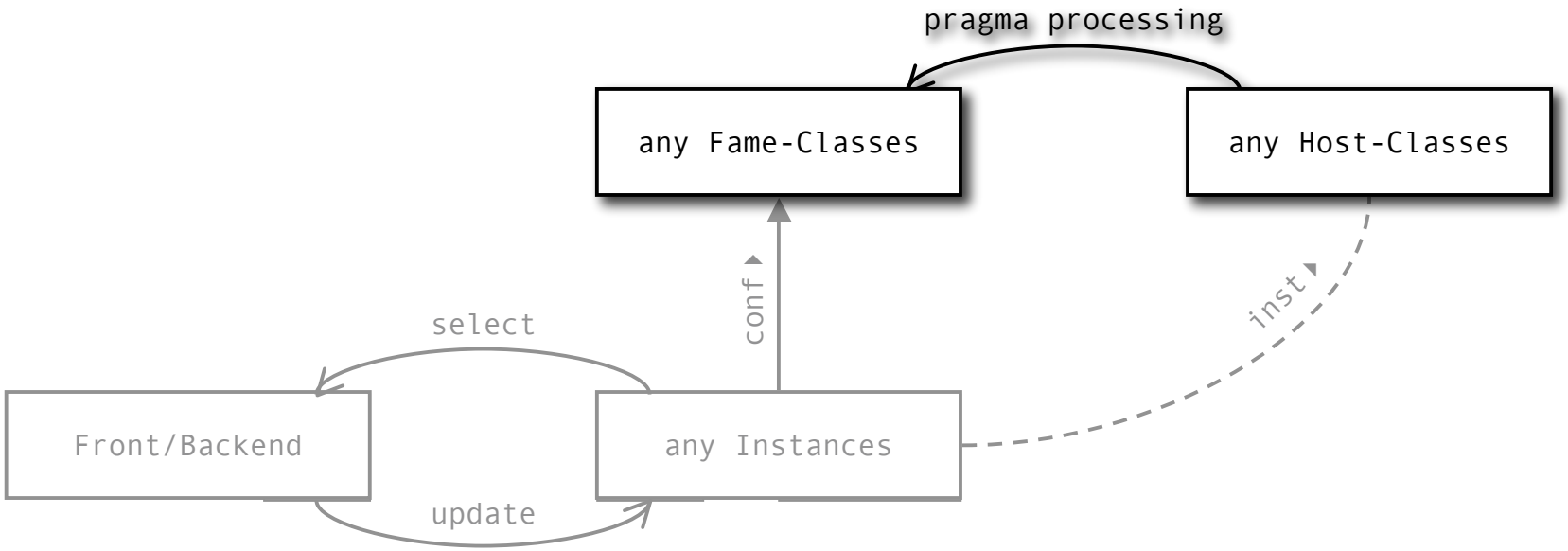


FM3 Meta-Metamodel

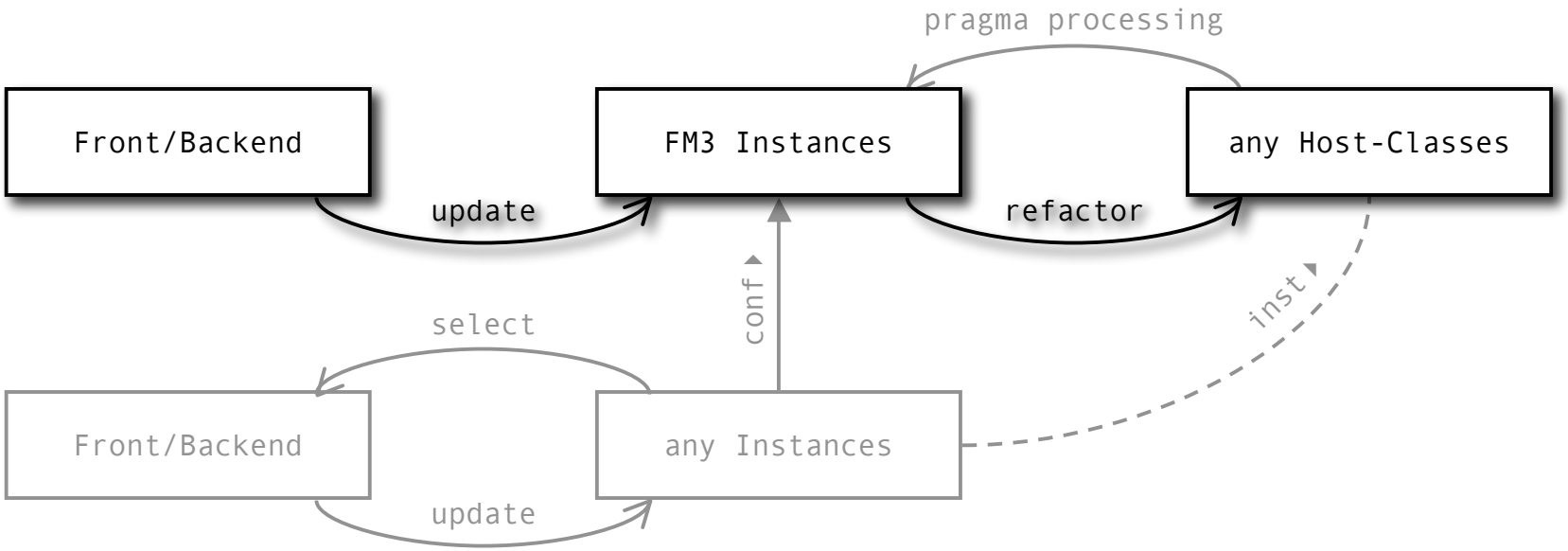
Fame- and Smalltalk classes are causally connected.



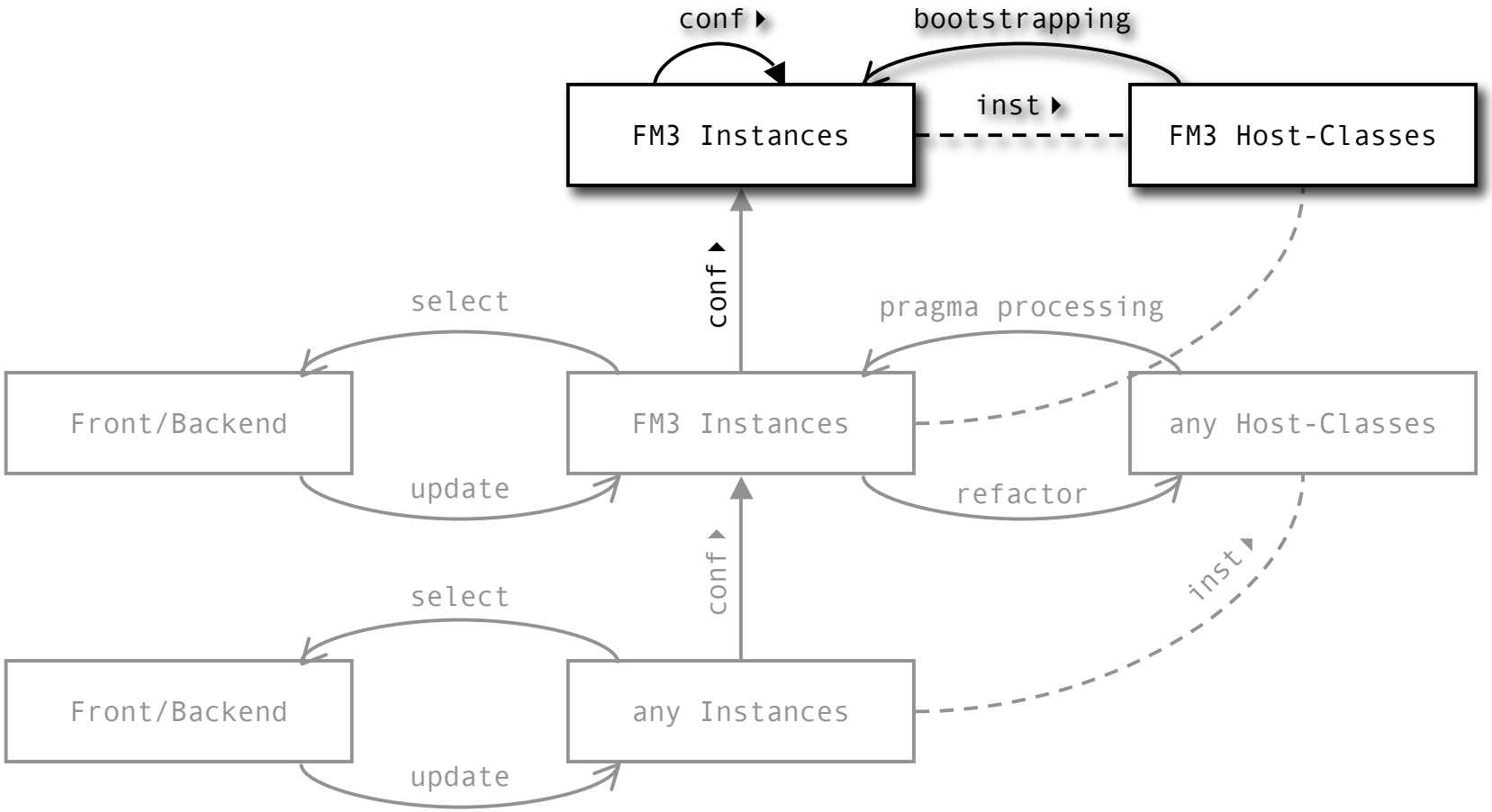
Runtime pragma processing makes the system **self-aware** of its structure.



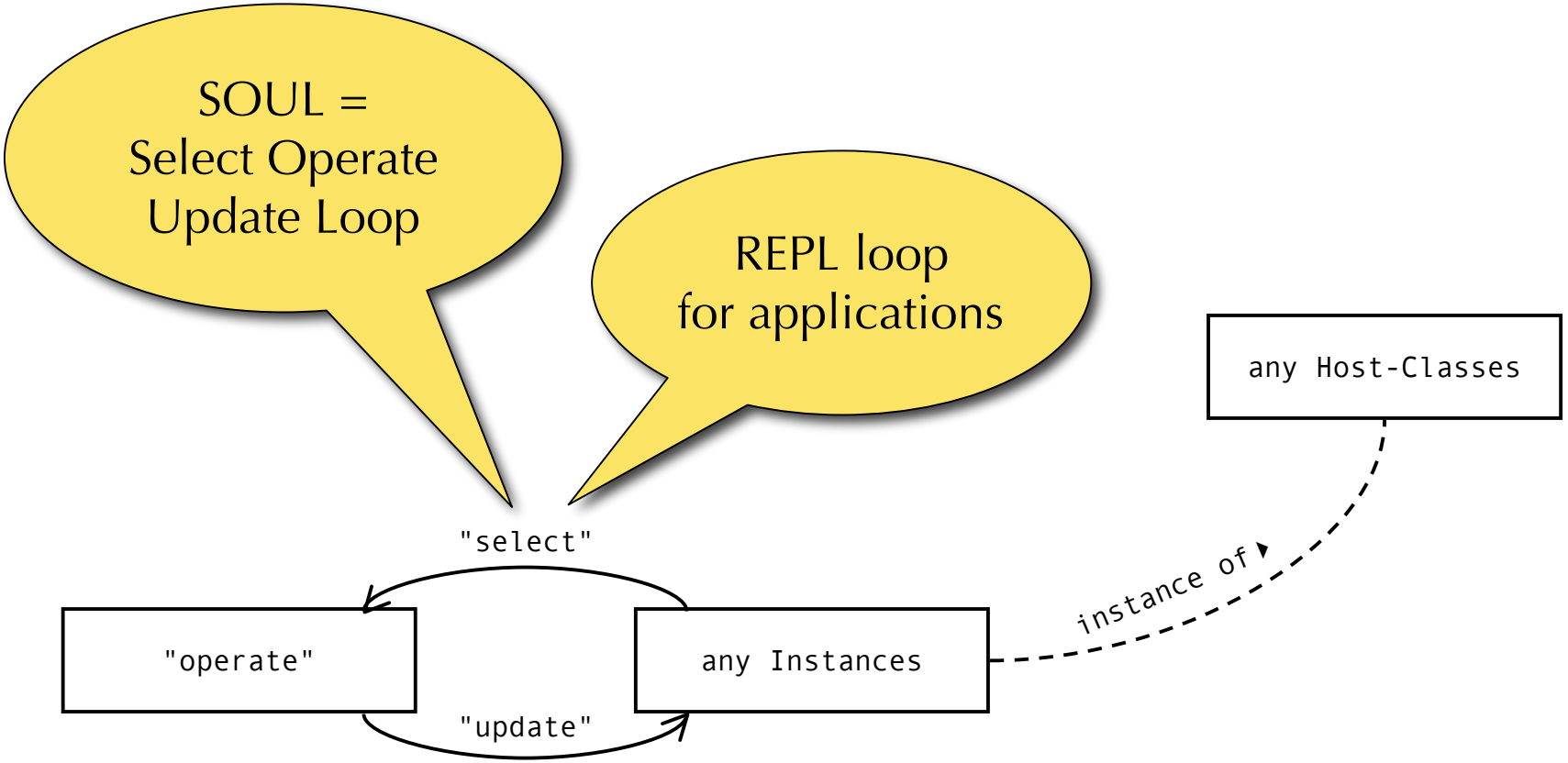
The system **adapts** to new data structures using runtime code generation.



The top-most layer is **self-described**.



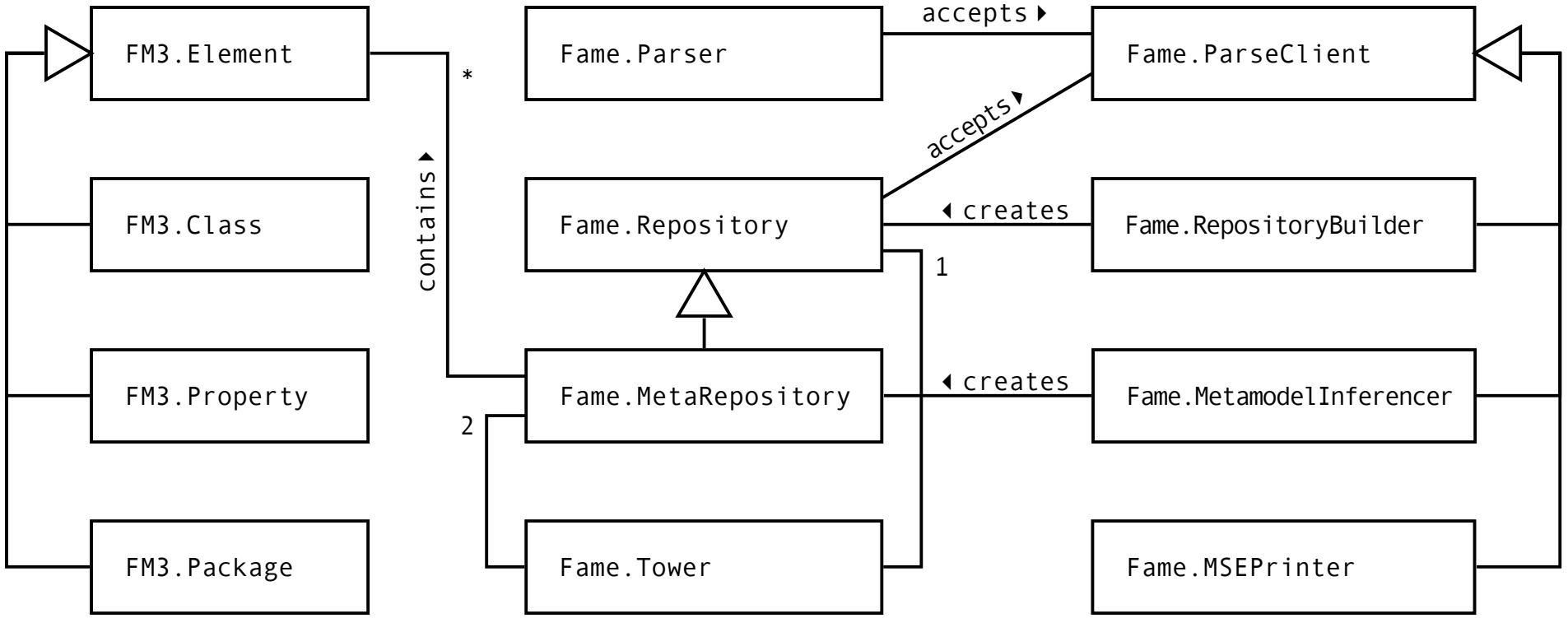
Do objects dream of virtual sheep?



DEMO

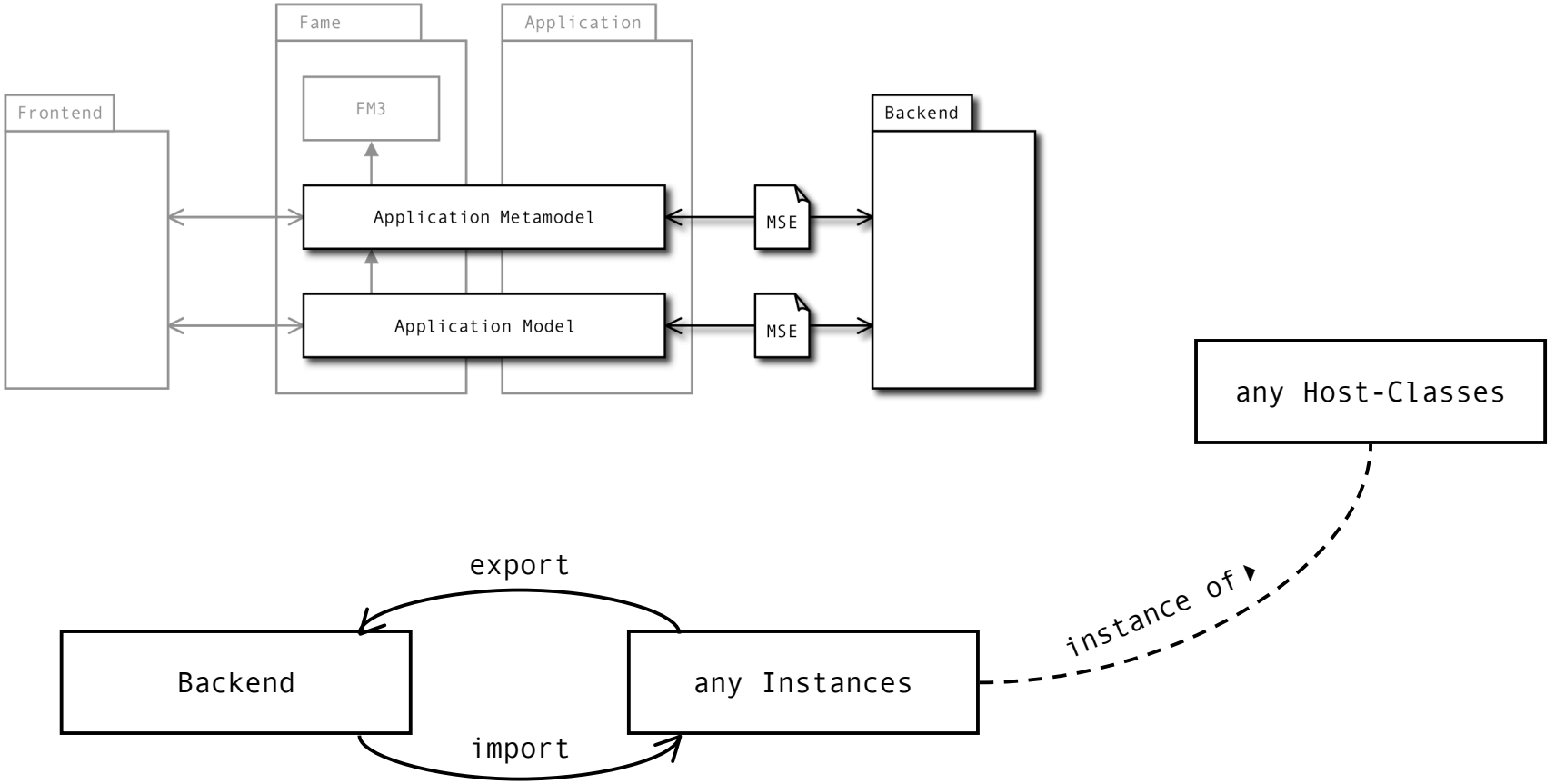
Nothing tops a running example.

Adrian Kuhn

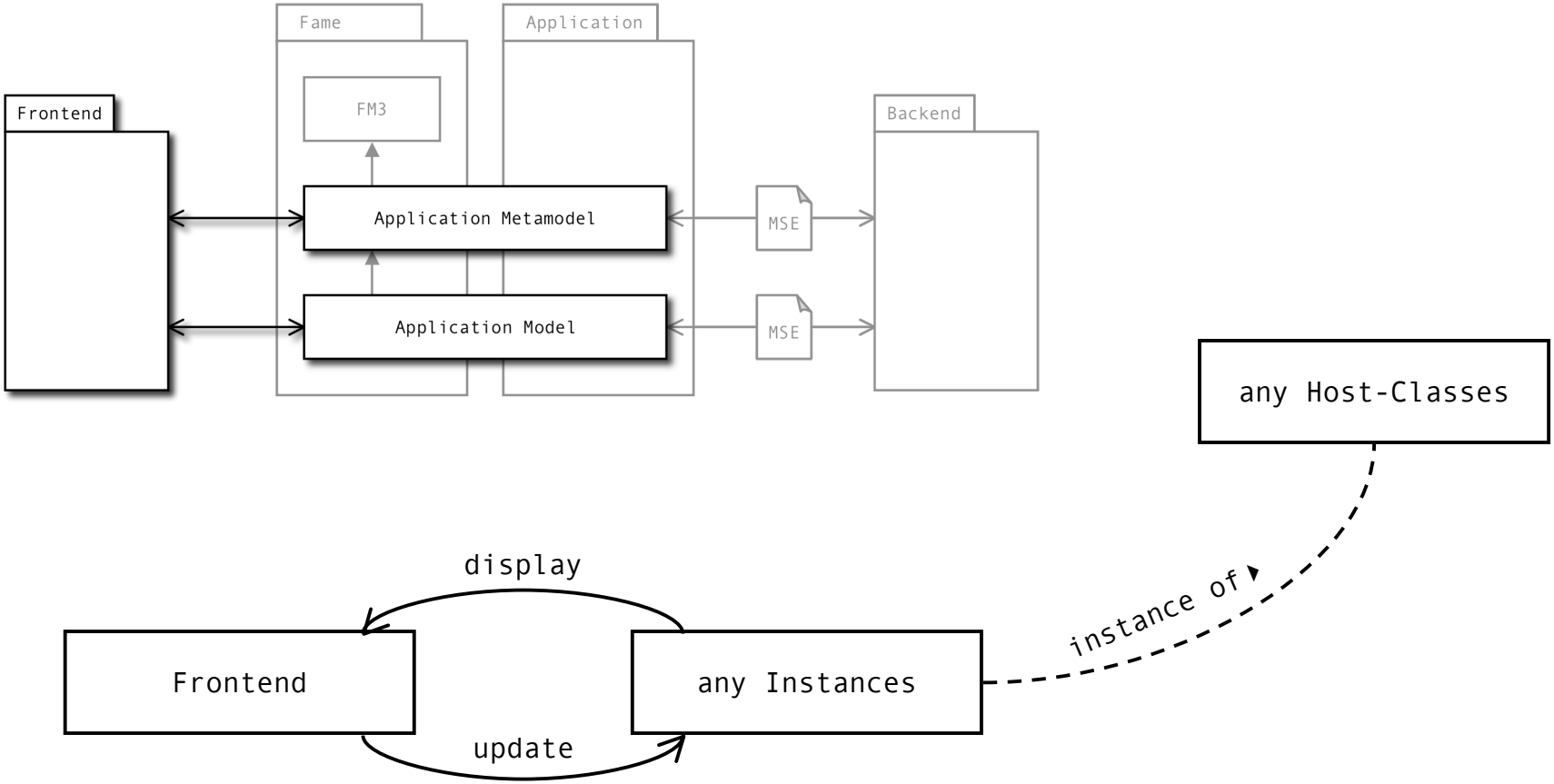


Core classes of the FAME framework.

MSE Files allow you to exchange any data.



The FAME API allows you to browse any data.



Extend your application at runtime.

FAME

Meta-Modeling at Runtime

Available at http://smallwiki.unibe.ch/fame

Adrian Kuhn