VariBru

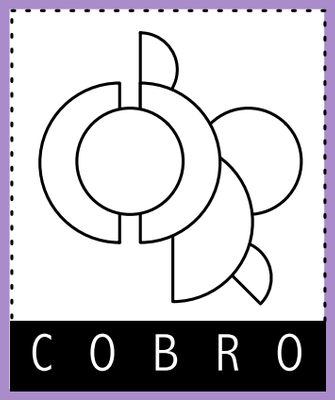
Variability in software-intensive product development

http://www.varibru.be

MoVES

Modelling, Verification and Evolution of Software

http://prog.vub.ac.be/moves



Even Better Flexibility in Smalltalk with the CoBro Runtime Domain Meta Layer

(version 2.9)

Dirk Deridder

Vrije Universiteit Brussel - System and Software Engineering Lab

http://ssel.vub.ac.be/dderidde

SVPP'08 Symposium

Vrije Universiteit Brussel, Brussels

August 8, 2008

Vrije Universiteit Brussel

CoBro: Research Context - Goals

To provide programming language support and tools
for the documented development of software systems
that can be *efficiently extended, adapted, or configured*
for use in a particular context

Many perspectives:

- Process perspective
- Modeling perspective
- Technology perspective

Short version:
Programmer support for software variability

- Software Product Lines, Frameworks, Domain Specific Languages, Macro's, Design Patterns, Polymorphism, ...

⇨ Our focus is on lightweight programming technology that doesn't impose an overall modus of operandi

Per Perspective - Many Dimensions

What should vary?

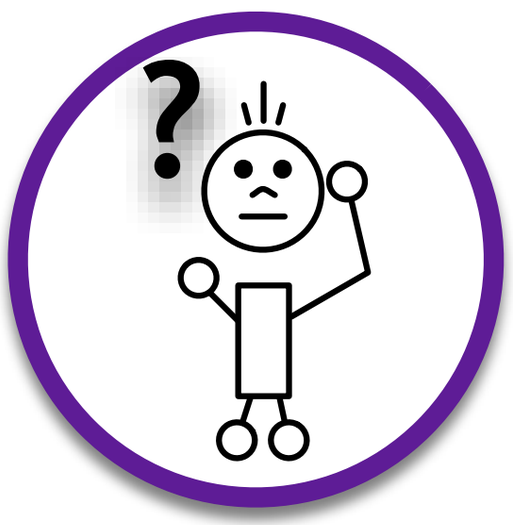
- Functionality (data, behavior, control flow,...)
- User Interface (context, user, task,...)
- Platform (hardware, OS, infrastructure,...)

When should it vary?

- Source-time (manual programming, generators,...)
- Compile-time (precompiler, aspect weaving,...)
- Link-time (static libraries, ...)
- Runtime (reflection,...)

Who should make it vary?

- Developer (configuration files, macro's,...)
- Domain expert (DSL, Meta-CASE tools,...)
- End-user (Preferences dialog,...)



...

So why do we need CoBro and not just...

Lightweight Programmer Techniques

- Use of standard OO techniques
  e.g. inheritance, polymorphism, delegation, metaprogramming
- Use of design patterns
  e.g. Abstract Factory, Strategy, ...
- Use of analysis patterns (proven generic domain-specific designs)
  e.g. Party, Portfolio, Quantity, Observation, ...

>> Programmer sets up own “meta” infrastructure
>> Rigid separation between “meta” and “base” levels
>> Generic code results in implicit domain knowledge
>> Connection of domain knowledge and code is lost
>> And what about the ‘efficiently’ in the definition?

my(‘Efficiently’) = ‘Clean’ + ‘Easy’ + ‘Digestible’ + ...

CoBro in a Nutshell

Provides a mechanism that makes it possible to *capture* domain knowledge in an explicit form

EXPLICIT

+

Provides a mechanism that makes it possible to *couple* this domain knowledge to an implementation

COUPLED

+

Provides a mechanism that makes it possible to *involve* domain knowledge actively to provide software functionality

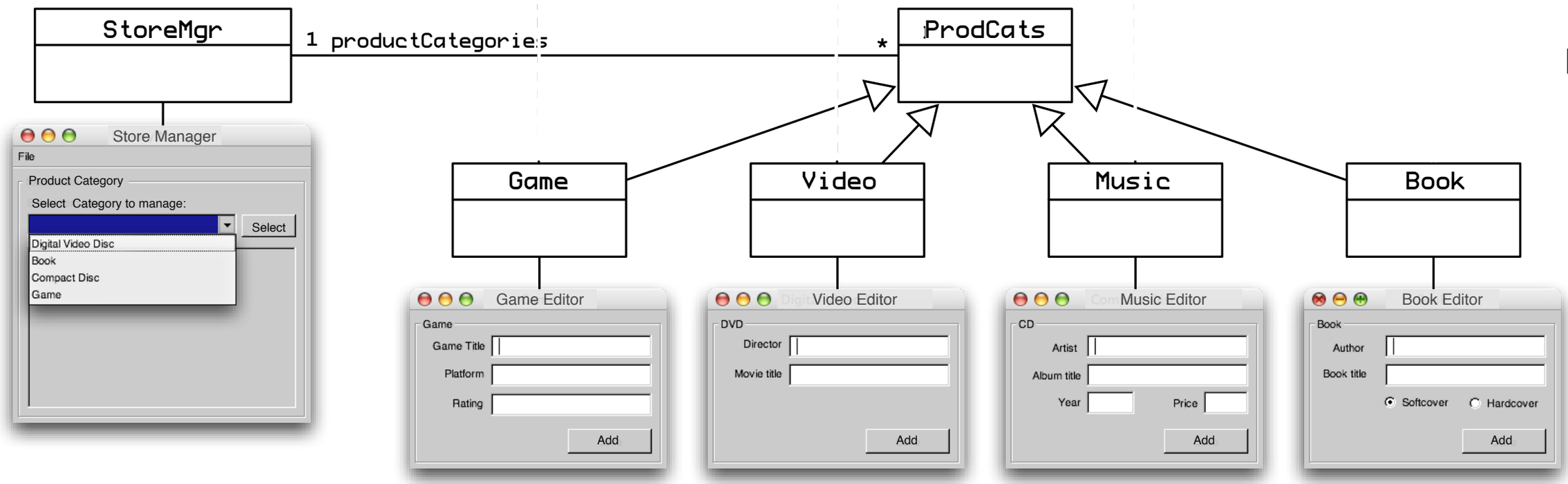
ACTIVE

+

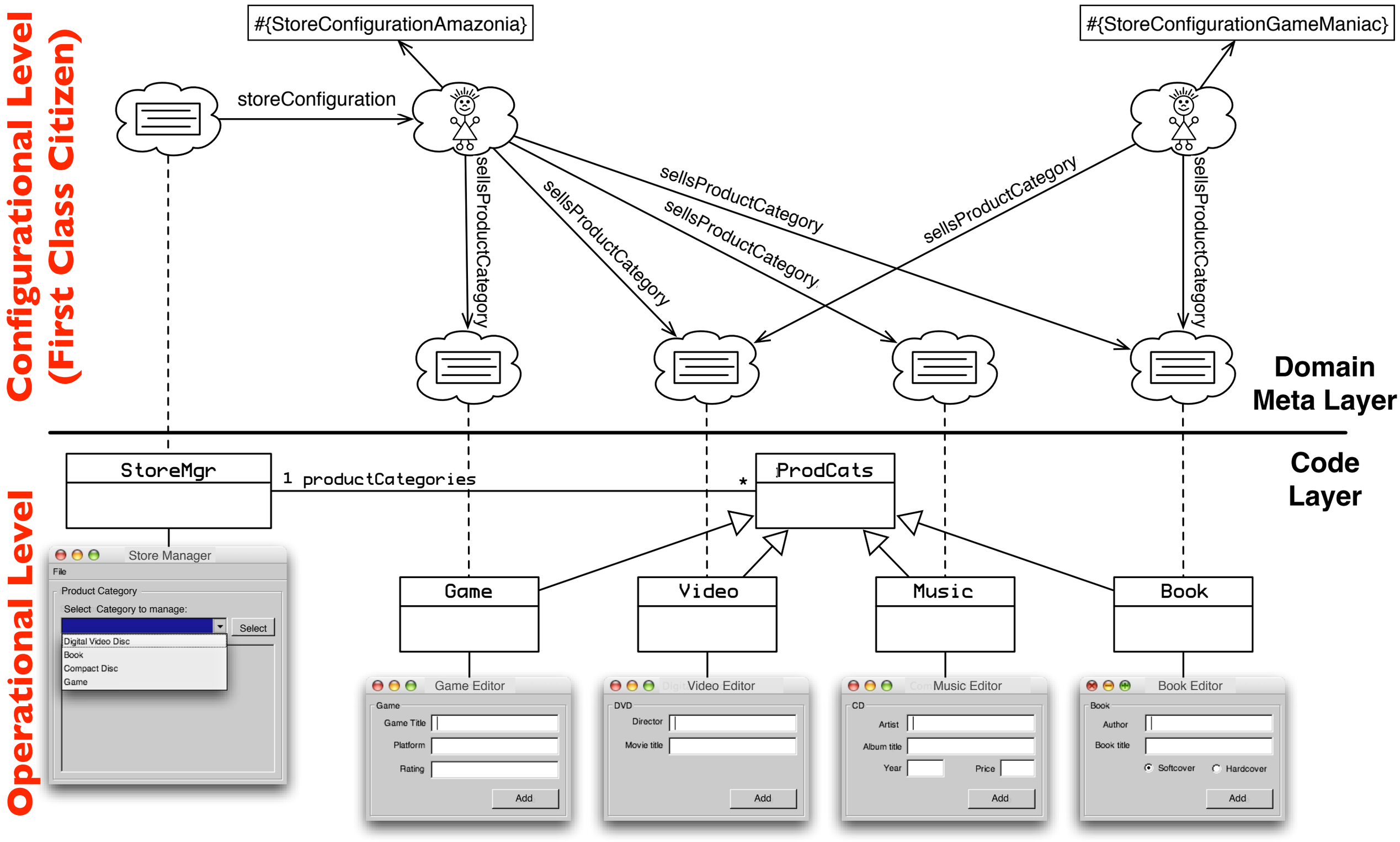
Provides a mechanism that makes it possible to *interact* with the domain knowledge in a transparent fashion

TRANSPARENT

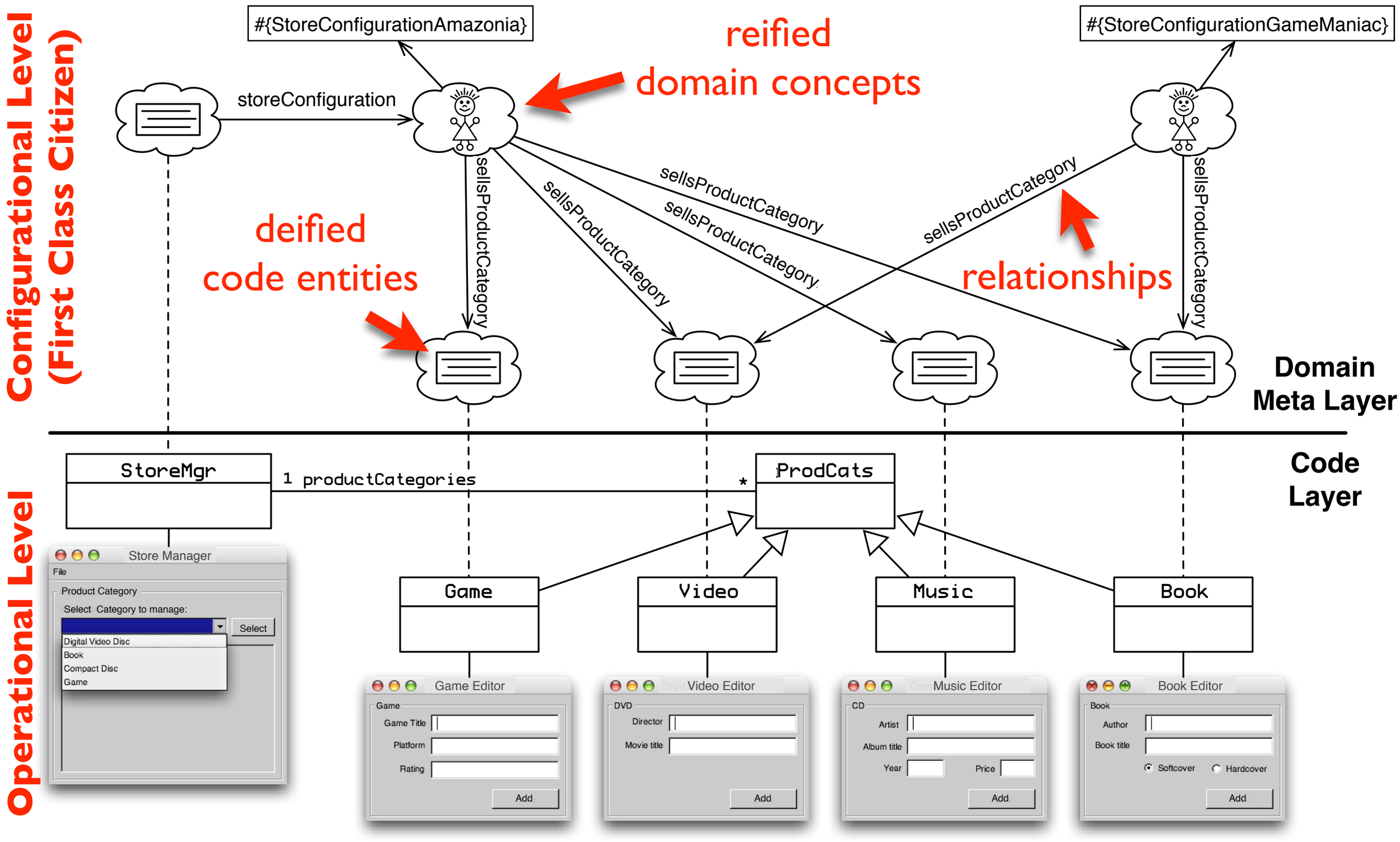
Active Domain Meta Layer?



Active Domain Meta Layer?



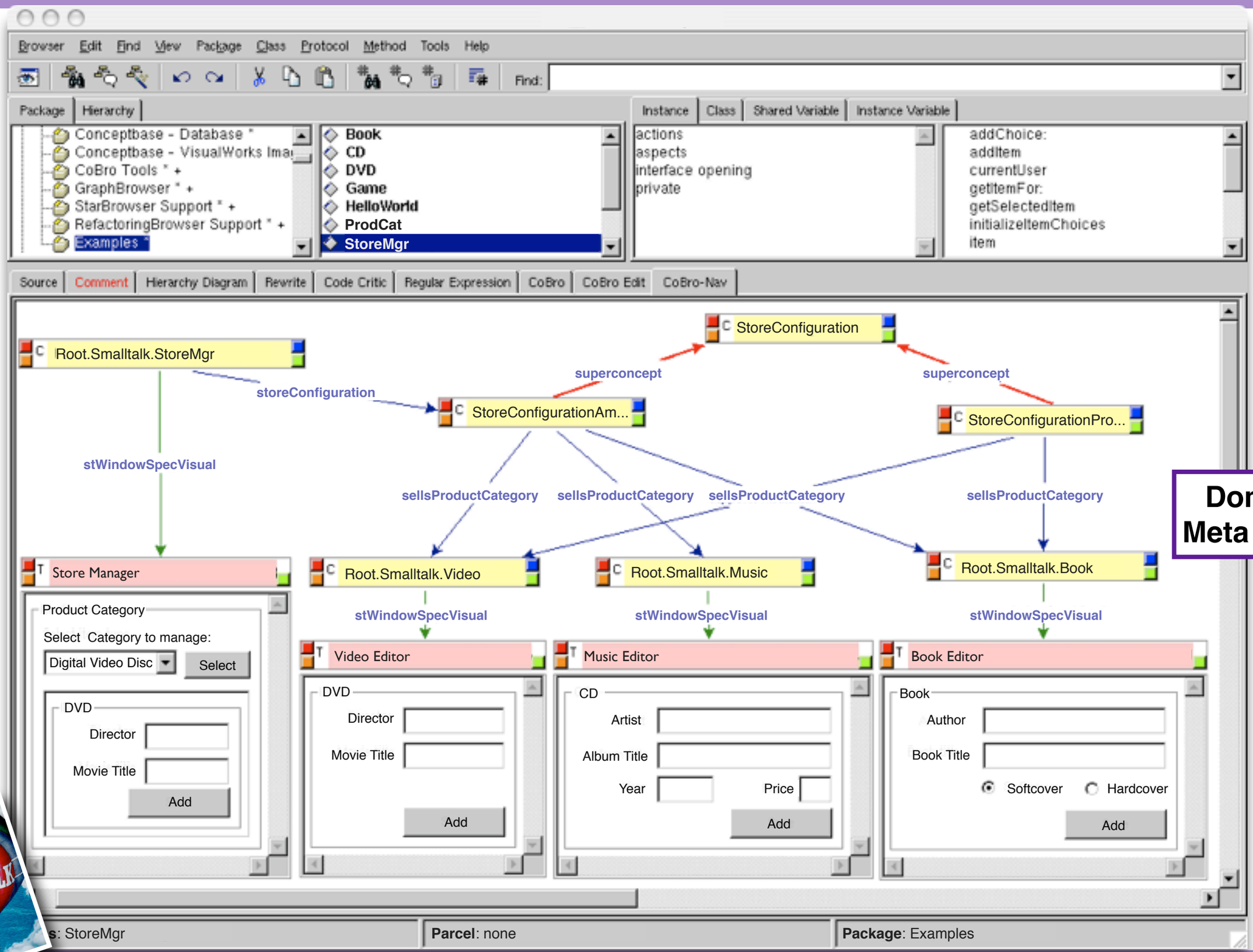
Active Domain Meta Layer?



Domain Meta Layer: Characteristics

- **Frame-based** KRS
  - a **concept** is defined by a definition frame
  - a **frame** is a collection of slots
  - a **slot** contains a relationship and a destination
  - a **destination** can be a concept or a terminal
  - a **terminal** can be any kind of object
    - e.g., String, Image, ST Blockclosure, ...
  - a **relationship** is also defined as a concept
- **Prototype-based**
  - everything is a concept (first-class)
  - parent slots for parent-chain lookup
    - multiple parents allowed
  - late/early binding of self
- (Partial) **Metacircular** implementation (bootstrapping)
- **Symbiosis** with Smalltalk and Close **Integration** with IDE

Close Integration Visualworks IDE



Domain Meta Layer

Close Integration Visualworks IDE

Build using CoBro to incorporate flexibility at several levels
(e.g., node visualisations, edge properties)

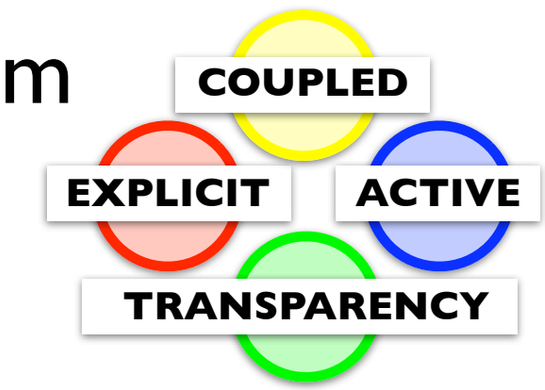
Domain Meta Layer

Symbiosis with Smalltalk (CML basics)

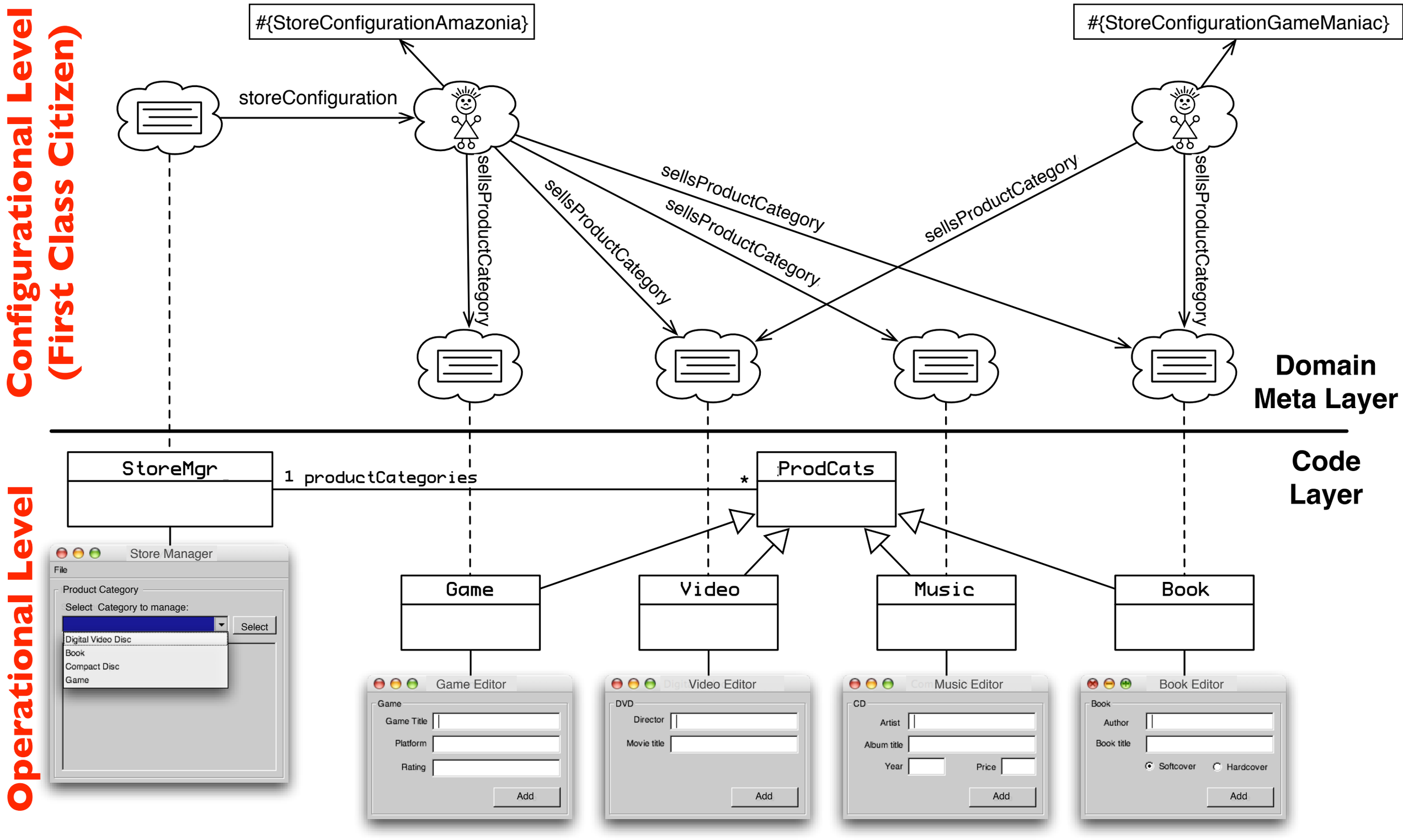
- **Concept creation**
  - Ex nihilo, editors, conceptification, cloning
  - Smalltalk syntax
  - Concept namespace + Extension of Smalltalk binding lookup

```
(Concepts defineConcept: #{HWorld}
    displayName: 'Hello World'
    superconcept: Concepts.Concept)
    comment: 'Testing CoBro' ;
    comment: 'This is fun'.
Concepts.HWorld comment.
Concepts.Number
  comment: 'Commenting a class'.
```

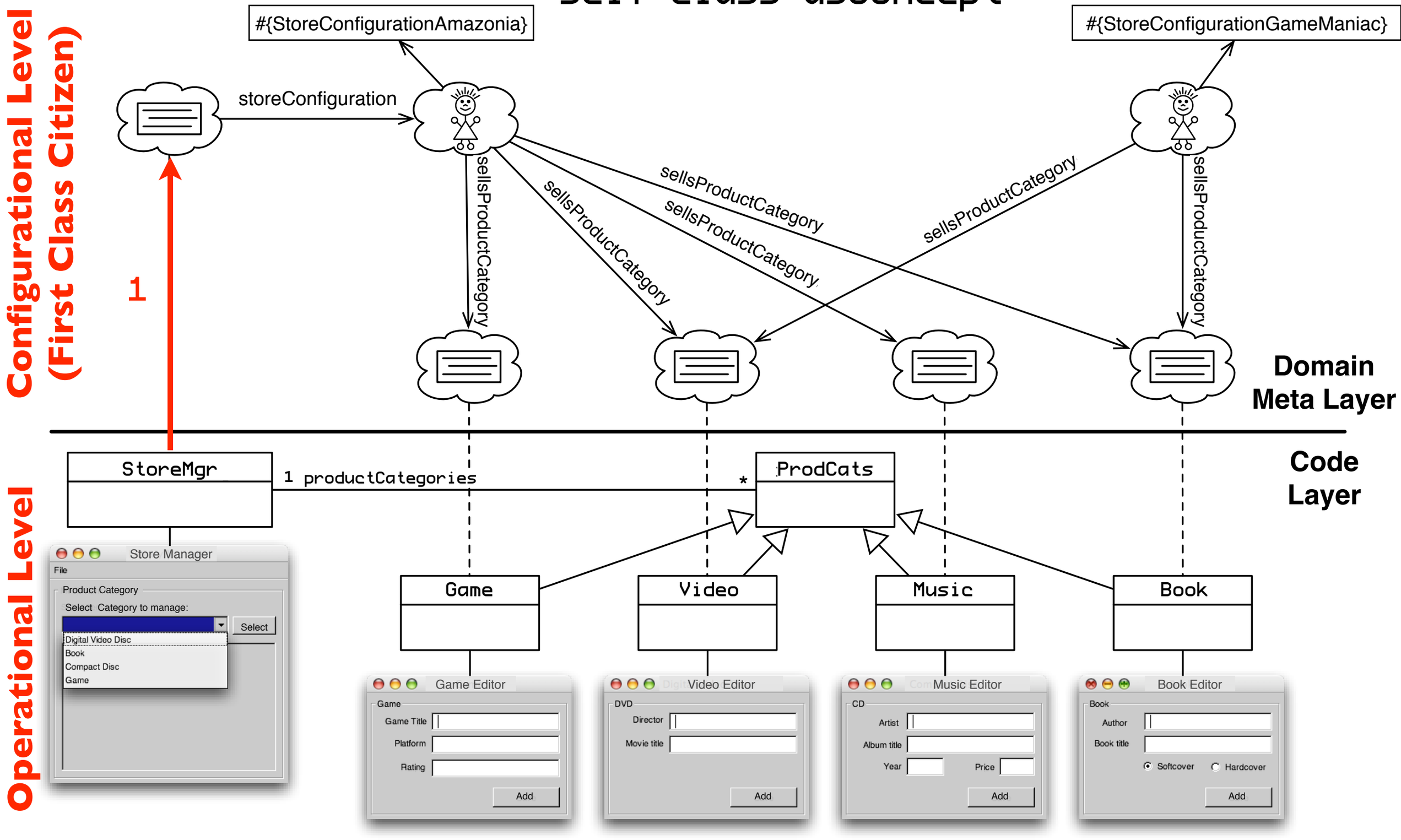
- **Concept manipulation**
  - Smalltalk message sending (symbiosis)
    - Message name = relation-concept name
  - Extension of Smalltalk method lookup mechanism
    - Parent lookup through 'superconcept' slots
  - CoBro Value Interpreters, Value Validators, ...



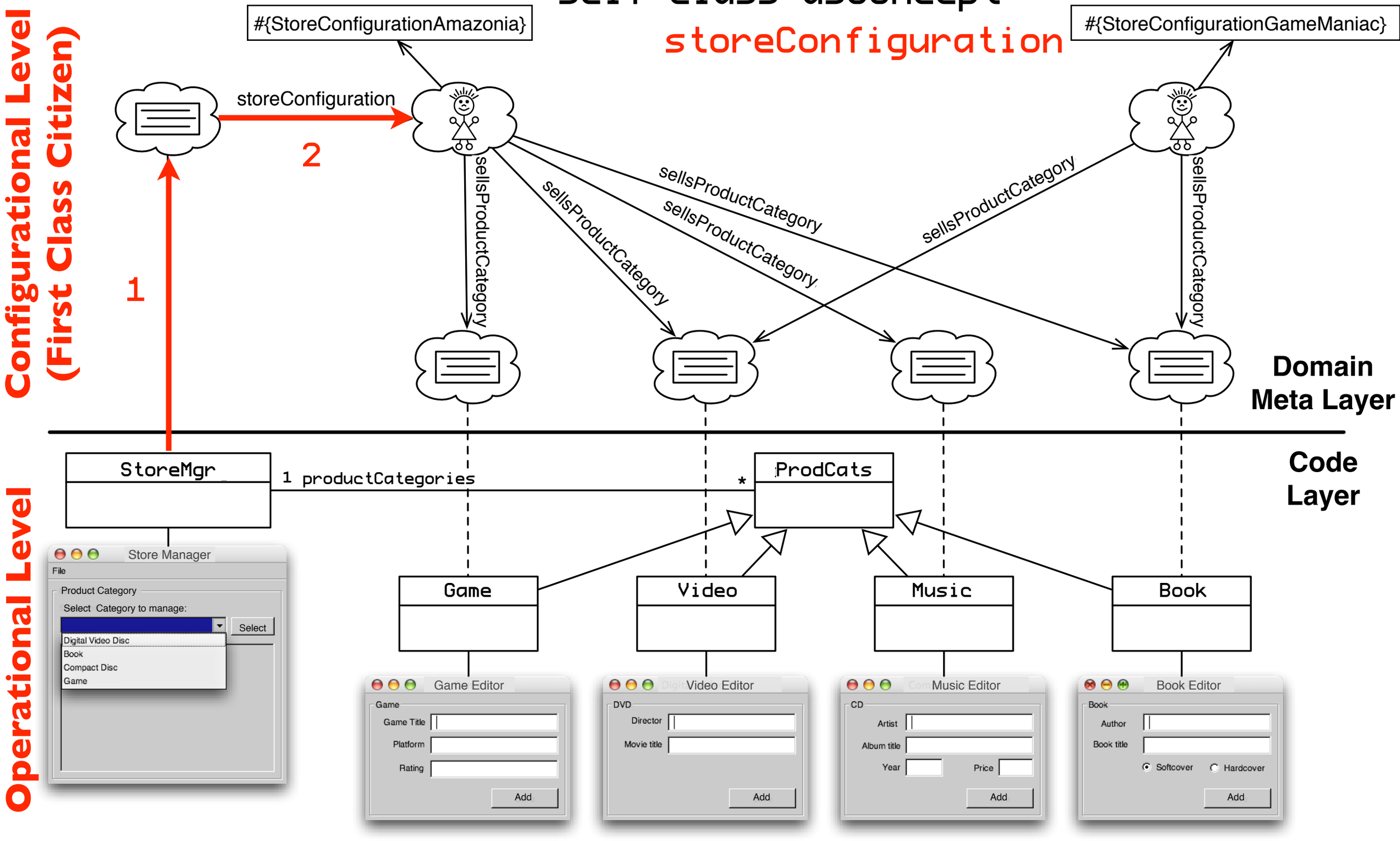
Active Domain Meta Layer in action



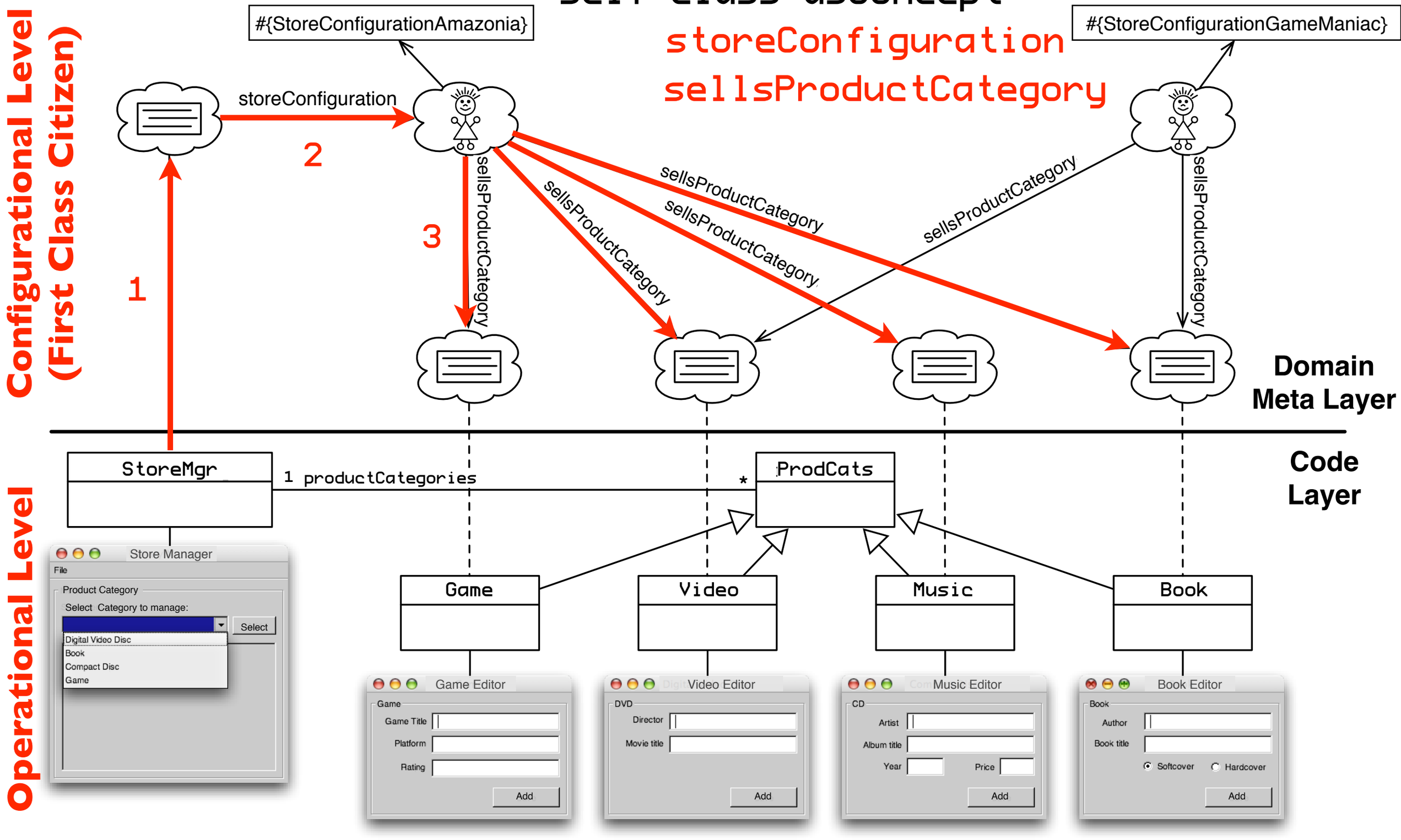
Active Domain Meta Layer in action



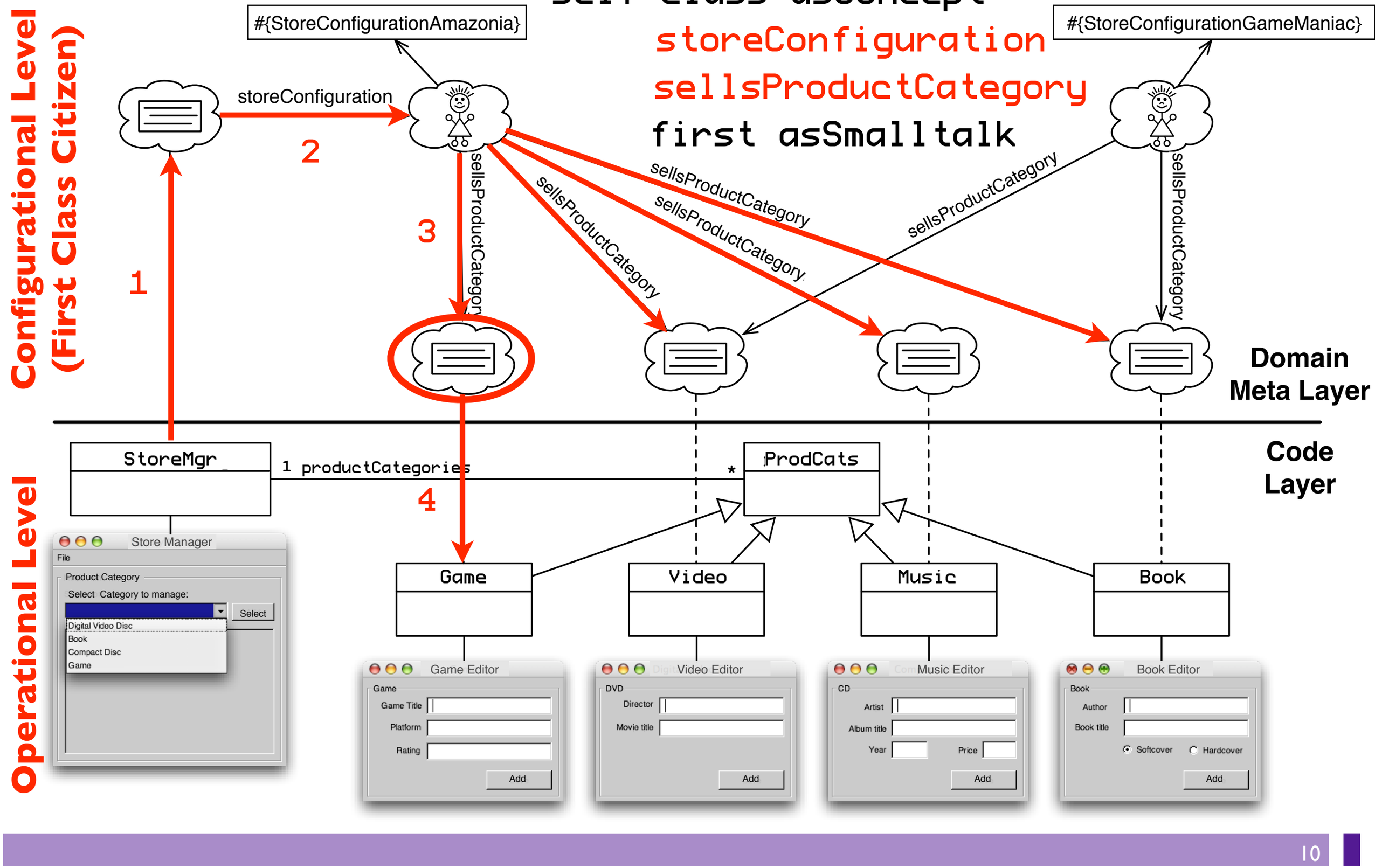
Active Domain Meta Layer in action



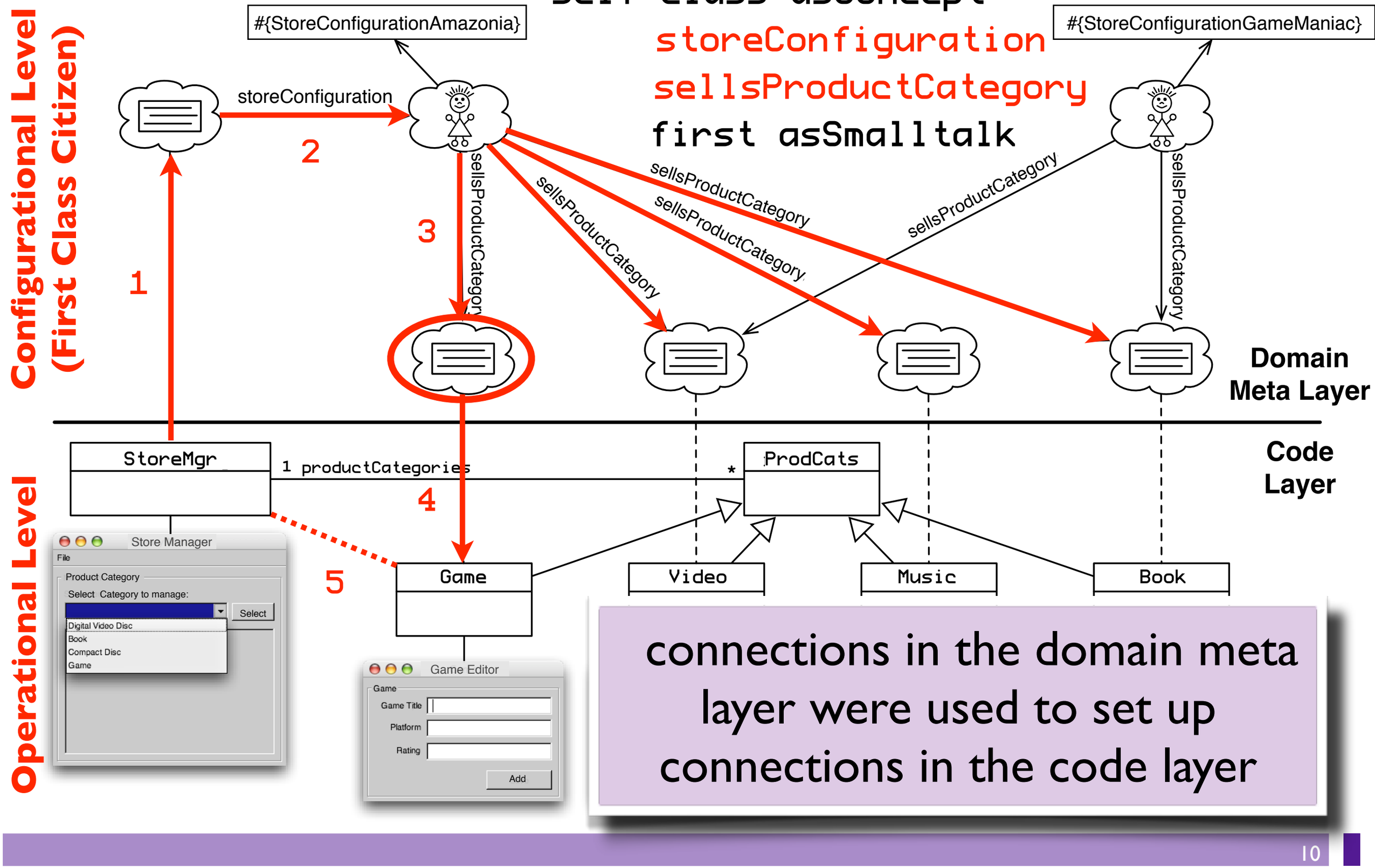
Active Domain Meta Layer in action



Active Domain Meta Layer in action

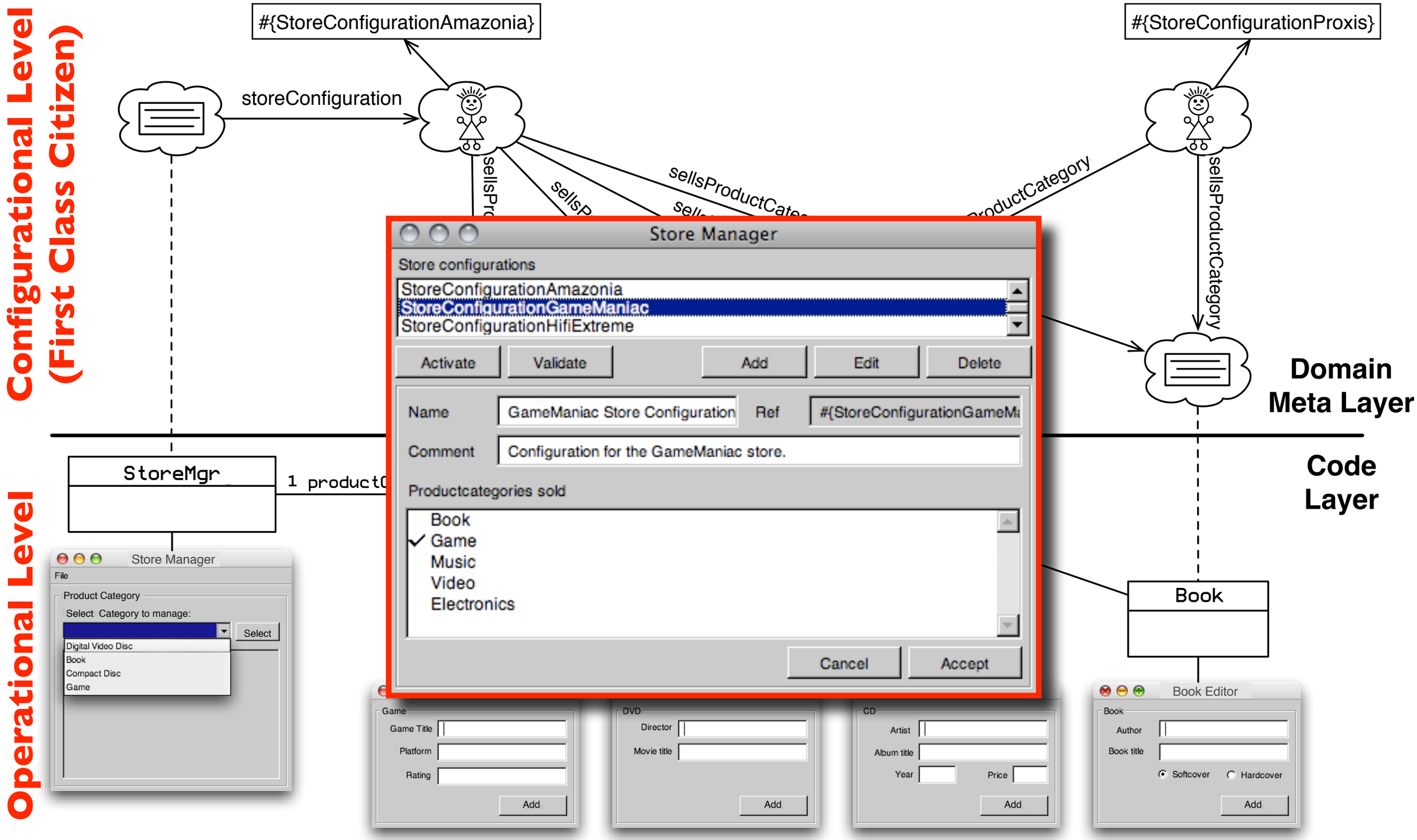


Active Domain Meta Layer in action

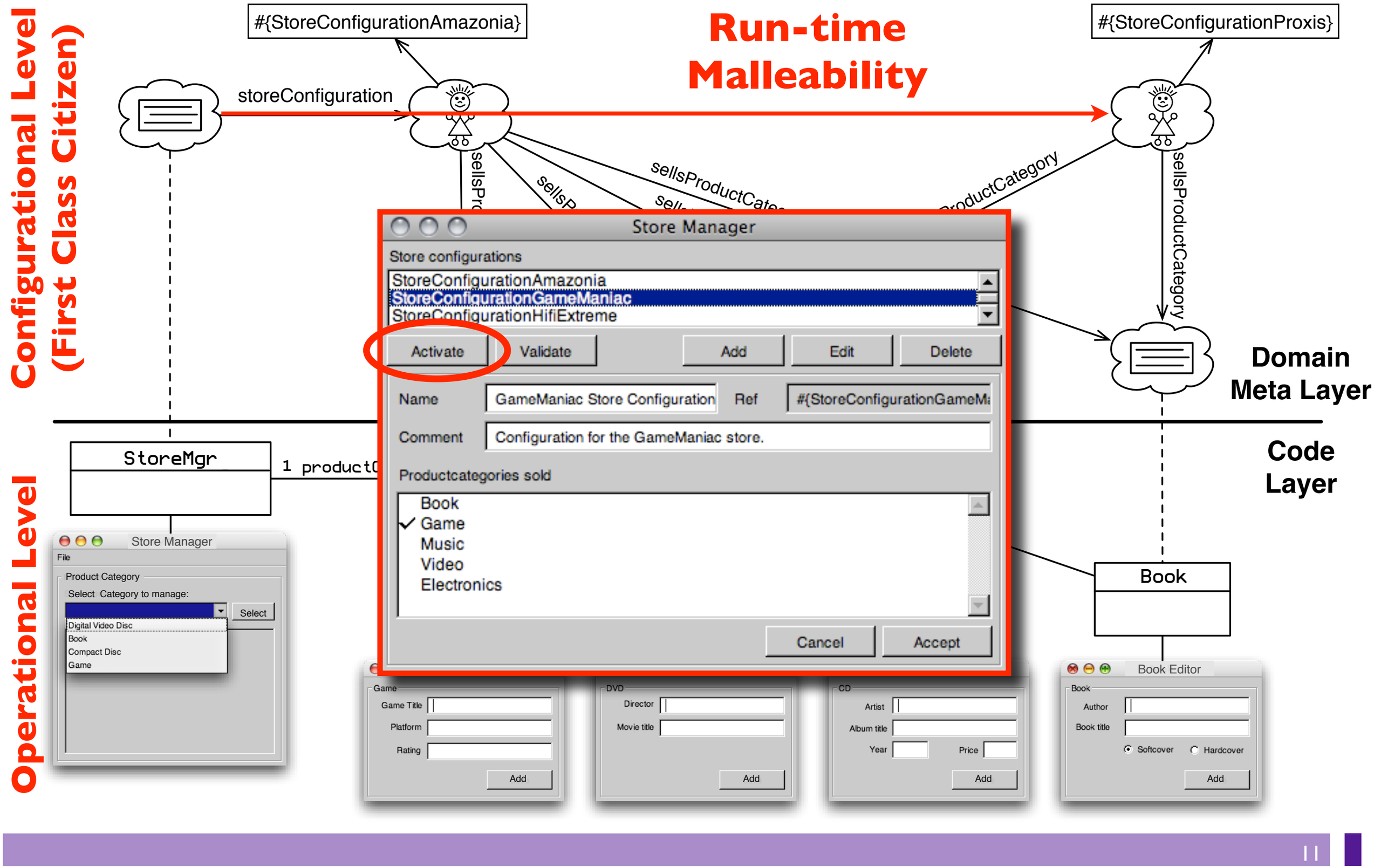


connections in the domain meta layer were used to set up connections in the code layer

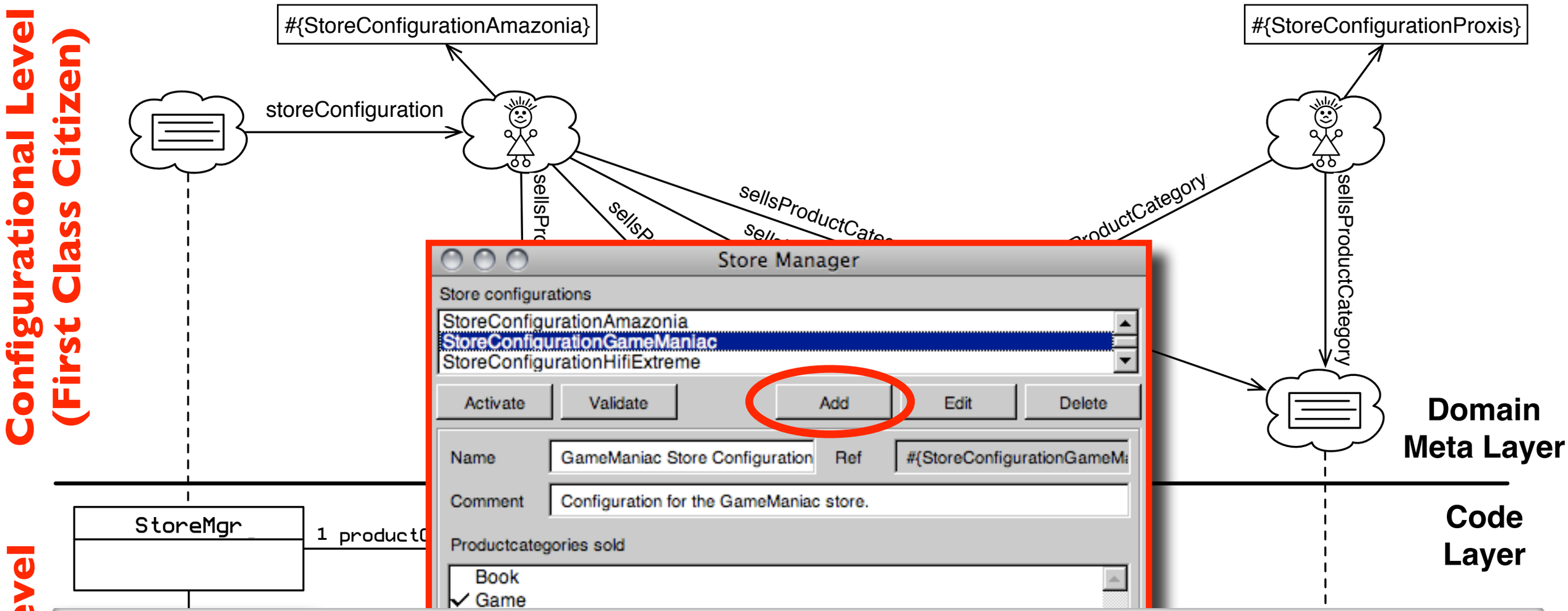
Active Domain Meta Layer



Active Domain Meta Layer



Active Domain Meta Layer

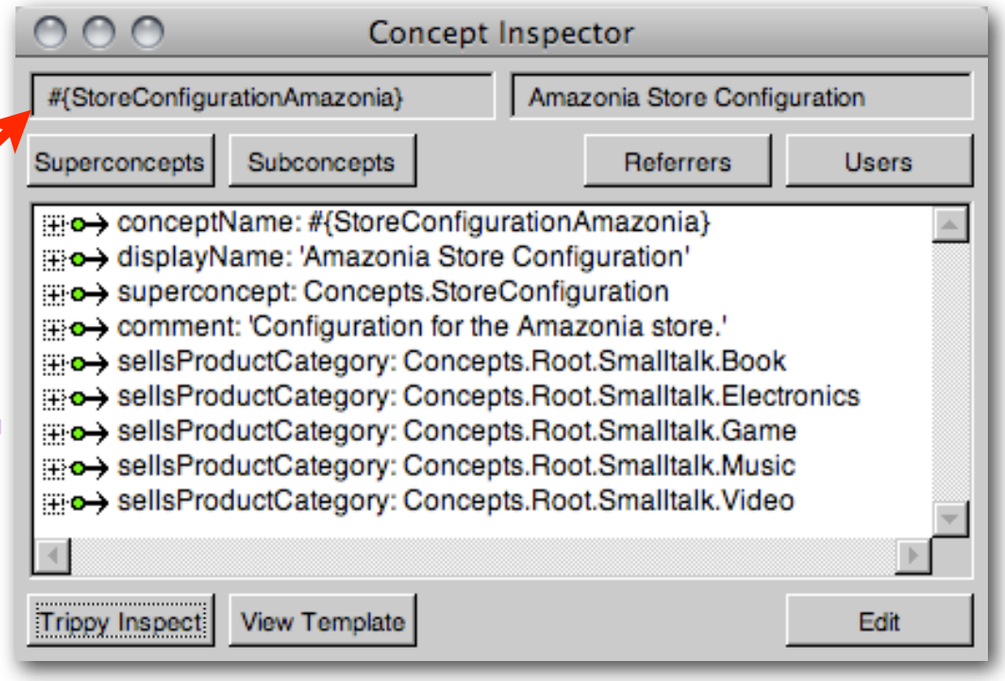



```
StoreManager>>addCategoryToConfig: aCategory
    |myConfig|
    myConfig := self class asConcept storeConfiguration.
    maxCategories := Concepts.sellsProductCategory multiplicity second.
    myConfig sellsProductCategory size <= maxCategories
        ifTrue: [ myConfig sellsProductCategory: aCategory ]
        ifFalse: [ Dialog warn: 'No more categories allowed !' ].
```

StoreConfiguration in CoBro-CML

```
(Concepts
    defineConcept: #{sellsProductCategory}
    displayName: 'sellsProductCategory'
    superconcept:Concepts.DomainRelationship)
    multiplicity: #(1 5) ;
    destinationType: Concepts.STClass.
```

```
(Concepts
    defineConcept: #{StoreConfigurationAmazonia}
    displayName: 'StoreConfigurationAmazonia'
    superconcept:Concepts.StoreConfiguration)
    sellsProductCategory: Concepts.Video ;
    sellsProductCategory: Concepts.Book ;
    sellsProductCategory: Concepts.Game ;
    sellsProductCategory: Concepts.Music ;
    sellsProductCategory: Concepts.Electronics.
```




```
StoreManager>>activateConfiguration
    StoreLauncher asConcept removeUsageOf:
                  Concepts.storeConfiguration.
    StoreLauncher asConcept
                  storeConfiguration: self selectedConfiguration
```

More Variability: ShippingRestrictions

International Shipping

Items from Amazon Marketplace can be shipped to several international regions, but cannot be shipped to **Africa, Island Nations, Israel, South America, or the Middle East.**

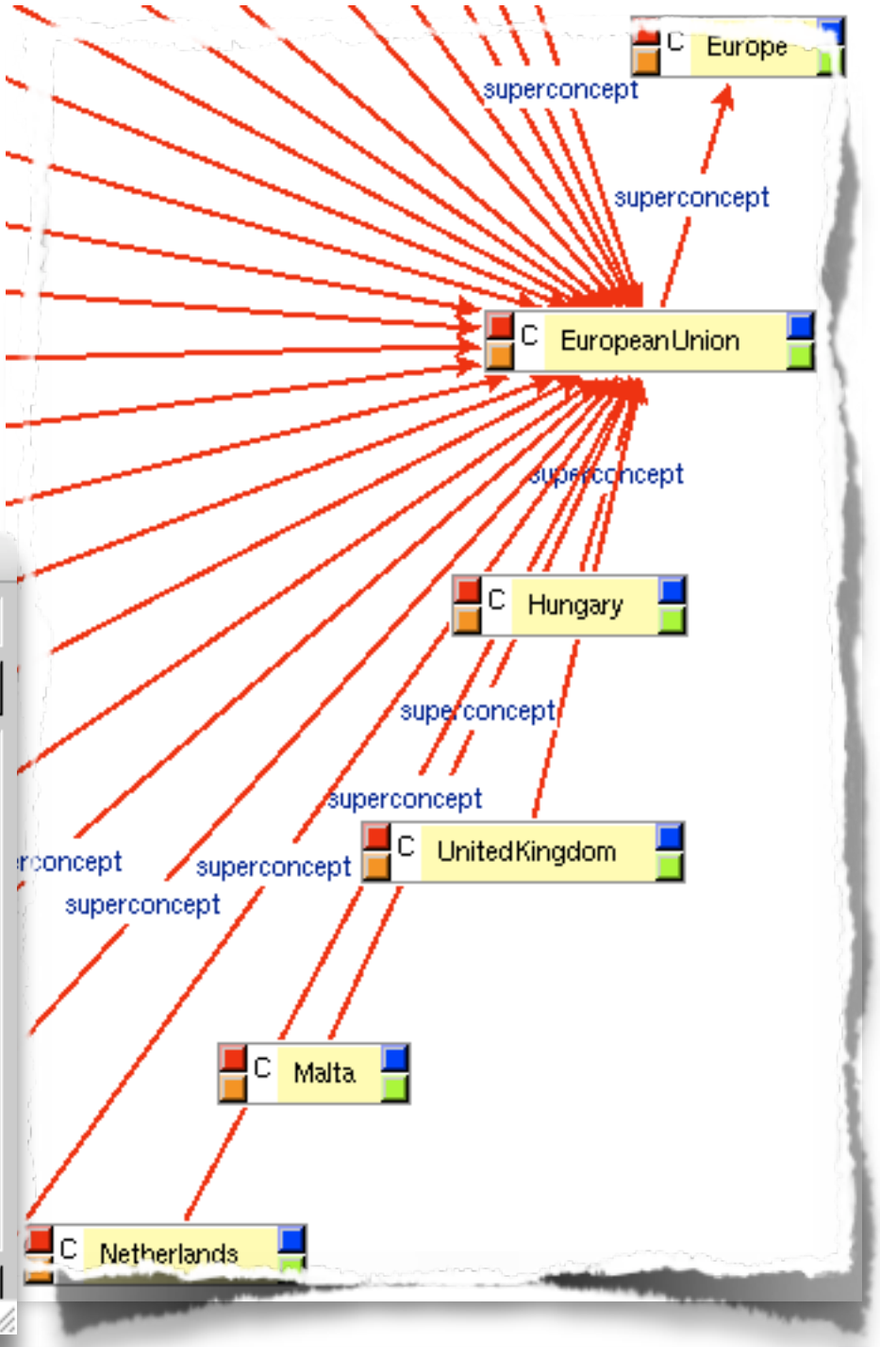
Read more about shipping to U.S. protectorates or APO/FPO addresses.

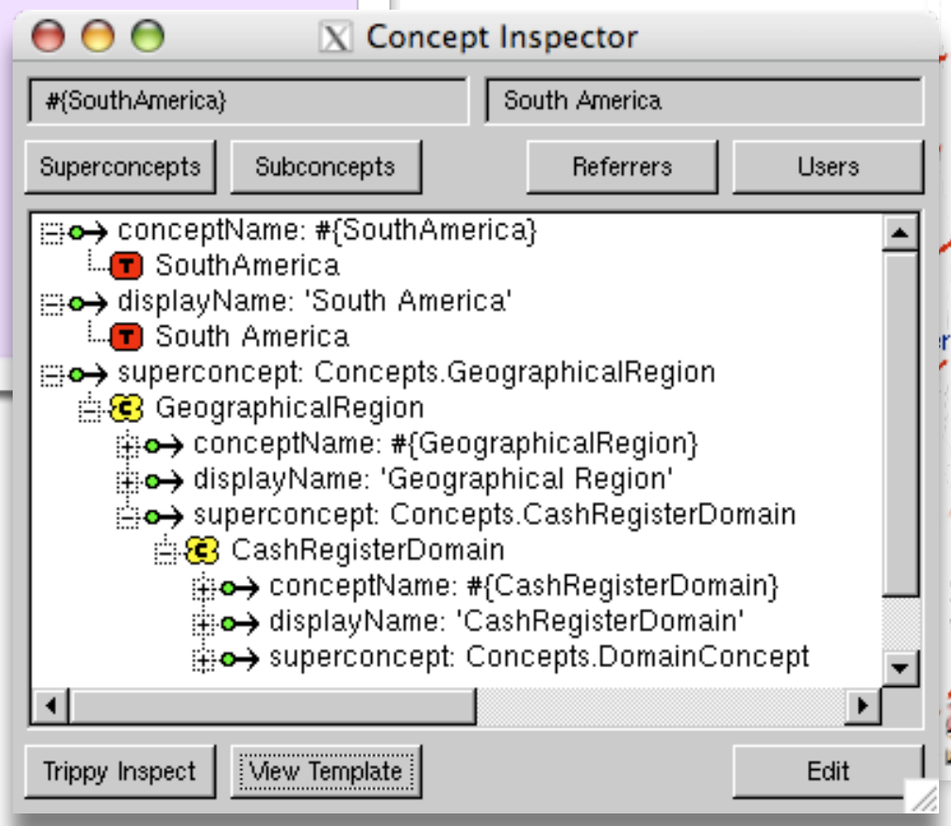
Restrictions

The following items can be shipped to almost all destinations outside the U.S.:

- **books***
- **DVDs**
- **music**
- **VHS videos**
- **...**

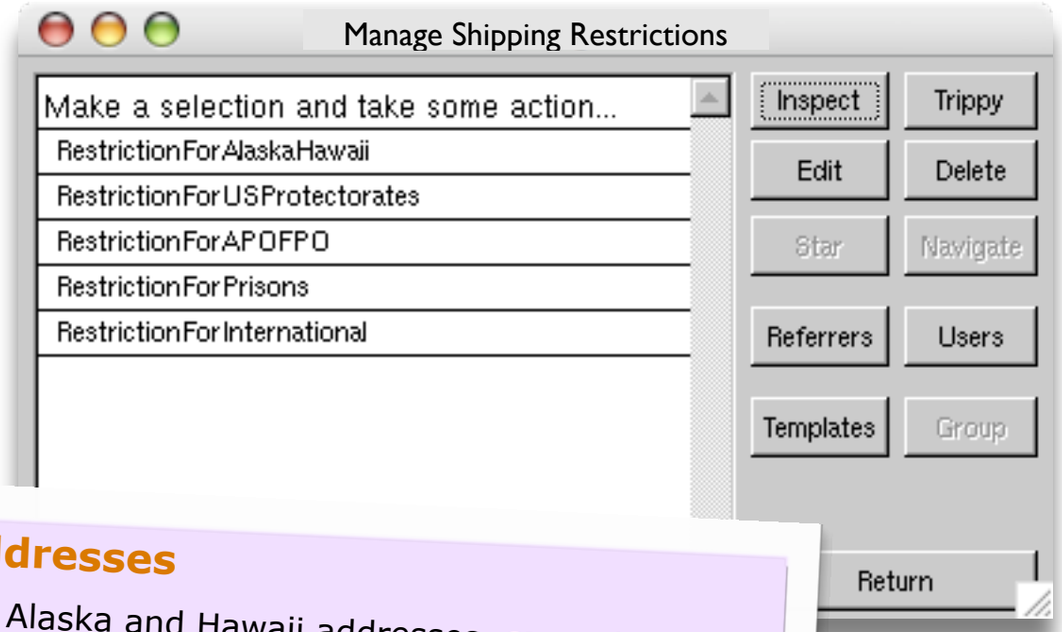
```
(Concepts
   defineConcept: #{SouthAmerica}
   displayName: 'South America'
   superconcept: Concepts.GeographicalRegion).
...
```





More Variability: ShippingRestrictions

```
(Concepts
   defineConcept: #{RestrictionForInternational}
   displayName: 'Shipping Restriction For International'
   superconcept: Concepts.ShippingRestriction)
   restrictedLocations: Concepts.Africa ;
   restrictedLocations: Concepts.SouthAmerica ;
   restrictedProductCategories: Concepts.Electronics ;
   comment: 'Items of our market place can be ...'.
```



U.S. Protectorates (including Puerto Rico)

Only the following items **can** be shipped to U.S. Protectorates:

- **baby items**
- **books**
- **DVDs**
- **music**
- **software**
- **toys**
- **VHS videos**
- **video games**

Alaska and Hawaii Addresses

Most items can be shipped to Alaska and Hawaii addresses, **except**:

- **grocery items**
- **gourmet food items**

...

Fun with Validators & Interpreters

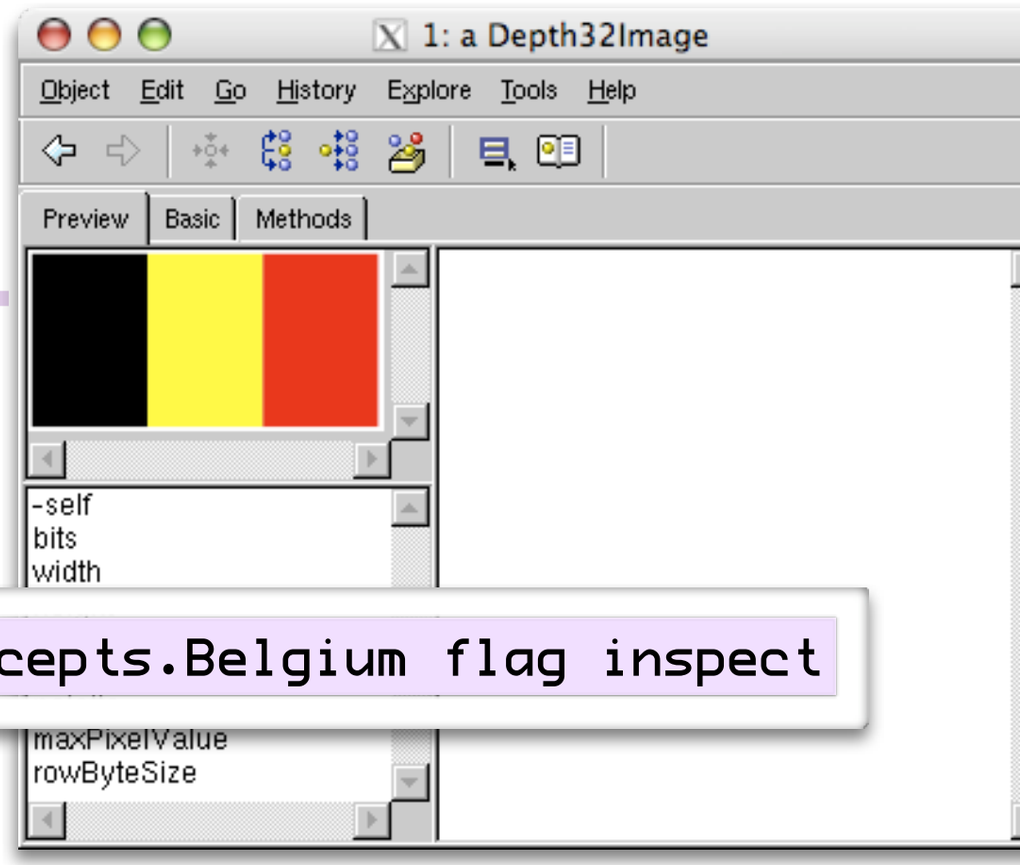
```
Concepts.StoreConfiguration
  configurationValidator:
    [:config |
       (config sellsProductCategory reject:
         [:each | each asSmalltalk
            inheritsFrom: ProdCat]) isEmpty]
```

can be used to provide default behavior in parent chain

```
(Concepts
   defineConcept: #{Belgium}
   displayName: 'Belgium'
   superconcept: Concepts.EuropeanUnion)
   flag: 'Flags/Flag_of_Belgium.png'.
```

```
(Concepts
   defineConcept: #{flag}
   displayName: 'flag'
   superconcept:Concepts.DomainRelationship)
   multiplicity: #(1 #n) ;
   destinationType: Concepts.STImage.
```

```
(Concepts defineConcept: #{STImage}
   displayName: 'STImage'
   superconcept: Concepts.SmalltalkTerminal)
   valueValidator: [:receiver :relation : destination|
                    destination asFilename exists] ;
   valueInterpreter: [:receiver :relation : destination|
                      (ImageReader fromFile: destination asFilename)
                         image].
```



CoBro has a rich MLI

Concept MLI

Namespace MLI

```
referrers
referrersUsing: aConcept
users
```

```
hasSuperconcept: aConcept
hasSuperconceptNamed: aName
```

```
conceptName
displayName
destinationType
multiplicity
valueInterpreter
valueValidator
```

```
referringTo: aConcept
referringTo: aConcept using: aRelationship
referringToStringPattern: aStringPattern
```

```
subconcept
superconcept
allSubconcepts
withAllSubconcepts
allSuperconcepts
withAllSuperconcepts
```

```
isNameAvailable: aName
suggestNameBasedOn: aName
```

```
bag
edit
inspect
navigate
starBrowse
trippyInspect

...
```

```
containsConcept: aConcept
containsConceptNamed: aName
containsConcepts: aCollection
containsConceptsNamed: aCollection
```

```
remove
removeSilently
removeReferencesTo: aConcept
removeUsageOf: aConcept
```

```
allConceptNames
allConcepts
allConceptSpaces
allRelationships
```

...

```
isConcept
isTerminal
isReferringTo: aConcept
isReferringTo: aConcept using: aConcept
isReferringToStringPattern: aStringPattern
isRelationship
isSmalltalkConcept
isUsing: aConcept
```

```
remove: aConcept
removeSilently: aConcept
```

```
defineConcept: aName displayName: aPrettyString superconcept: aConcept
defineTransientConcept: aName displayName: aPrettyString superconcept: aConcept
```

CoBro has a rich MLI

Concept MLI

Namespace MLI

```
referrers
referrersUsing: aConcept
users
```

```
hasSuperconcept: aConcept
hasSuperconceptNamed: aName
```

```
conceptName
displayName
destinationType
multiplicity
valueInterpreter
valueValidator
```

```
referringTo: aConcept
referringTo: aConcept us
referringToStringPattern
```

```
isNameAvailable: aNam
suggestNameBasedOn: a
```

```
containsConcept: aConcep
containsConceptNamed: aN
containsConcepts: aColle
containsConceptsNamed: a
```

```
allConceptNames
allConcepts
allConceptSpaces
allRelationships
```

```
remove: aConcept
removeSilently: aConcept
```

Build using CoBro to incorporate flexibility
(partially metacircular)

So you get flexibility to play with:
- parent chain lookup
- code entity deification
- consistency validators
- ...

```
bag
edit
inspect
navigate
starBrowse
trippyInspect
...
```

```
sing: aConcept
n: aStringPattern
isSmalltalkConcept
isUsing: aConcept
```

```
defineConcept: aName displayName: aPrettyString superconcept: aConcept
defineTransientConcept: aName displayName: aPrettyString superconcept: aConcept
```

Future Work

- Feature Diagrams to model the variability
  - Extension to the graphical navigator
- Continue to exploit / explore the power of valueValidators & valueInterpreters
- Provide extensive built-in consistency management and co-evolution support
- Establish "Methodology"-side of C3
  - Guidelines, Patterns, Considerations, ...
- Open question:
  - How to check if the variability is indeed supported "efficiently"? (Quality Measures)

Vrije Universiteit Brussel

Faculty of Sciences

Department of Computer Science

System and Software Engineering Lab

Dirk Deridder

Pleinlaan 2

B-1050 Brussel

Belgium

Office 4K208

Campus Etterbeek - Building K

Tel : +32 2 629 29 65

Fax : +32 2 629 28 70

Dirk.Deridder@vub.ac.be

http://ssel.vub.ac.be/dderidde/