

MOVES

Modelling, Verification and Evolution of Software

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

Supporting Functional Software Variability with Concept-Centric Coding (C3)

Dirk Deridder

Vrije Universiteit Brussel - System and Software Engineering Lab

<http://ssel.vub.ac.be/deridde>

BENEVOL 2007 - PRECISE - Namur

December 13-14, 2007

Functional Software Variability

Functional Software Variability

The ability of a software system or artefact to be *efficiently* extended, changed, customised or configured for use in a *particular context*

[Svahnberg et al. 2005]

Functional Software Variability

The ability of a software system or artefact to be *efficiently* extended, changed, customised or configured for use in a *particular context*

[Svahnberg et al. 2005]

➔ Many Perspectives:

- Managerial/Engineering Perspective
- Modeling Perspective
- Technology Perspective

Functional Software Variability

The ability of a software system or artefact to be *efficiently* extended, changed, customised or configured for use in a *particular context*

[Svahnberg et al. 2005]

➔ Many Perspectives:

- Managerial/Engineering Perspective
- Modeling Perspective
- Technology Perspective

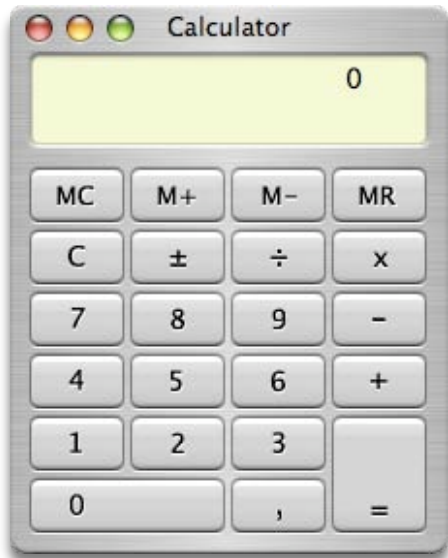


For a programmer the essential question is:

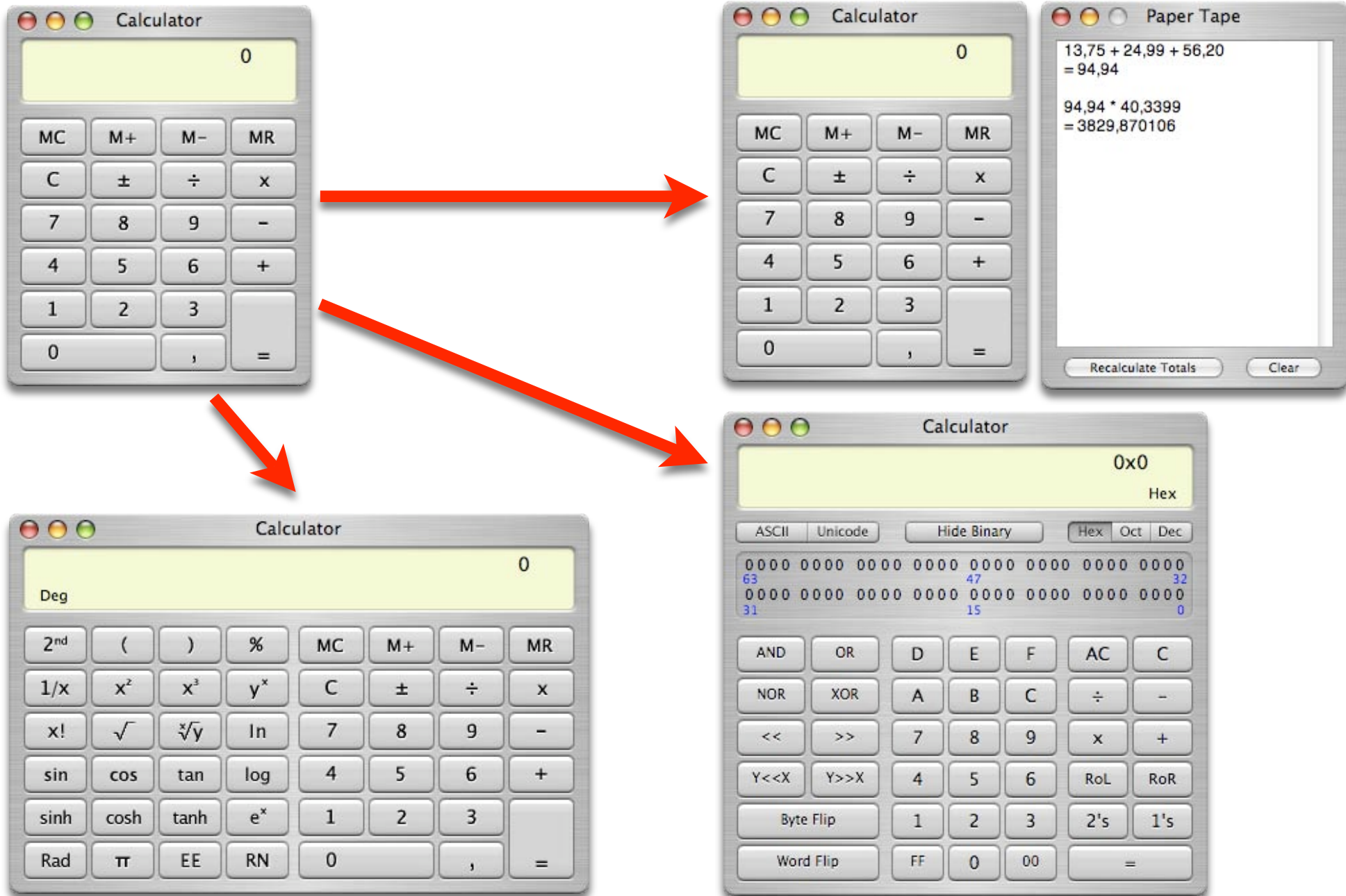
How to write flexible software (efficiently)?

- Software Product-Lines, Frameworks, DSLs, Programming Language Support, ...

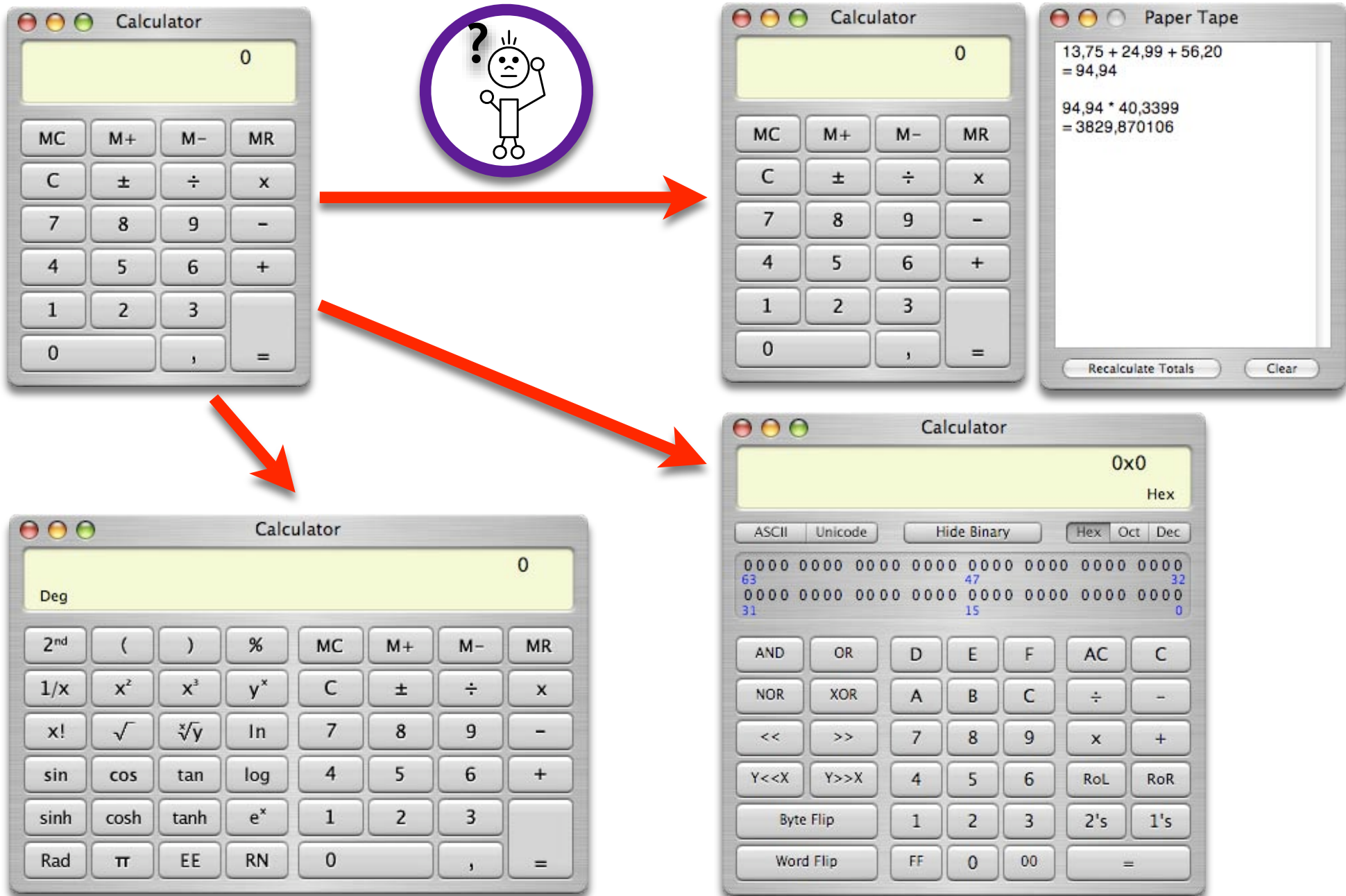
A Variability Example...



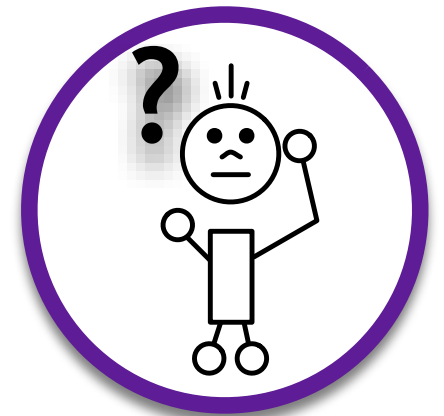
A Variability Example...



A Variability Example...



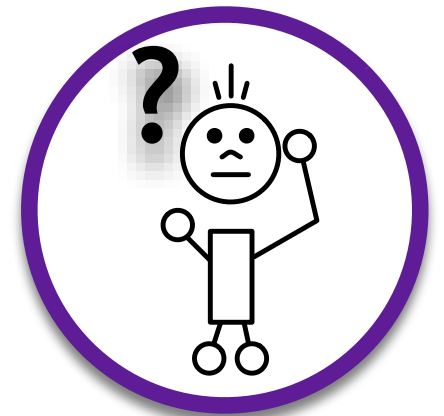
Per Perspective - Many Dimensions



Per Perspective - Many Dimensions

What should vary?

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



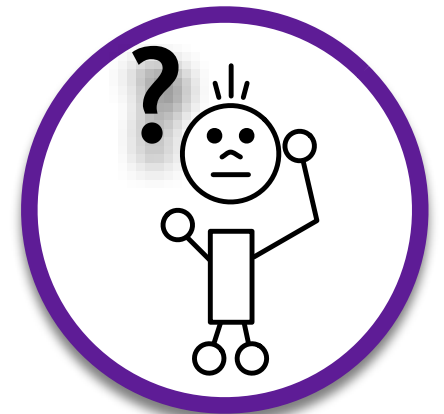
Per Perspective - Many Dimensions

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)
- Runtime (reflection), ...



Per Perspective - Many Dimensions

What should vary?

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

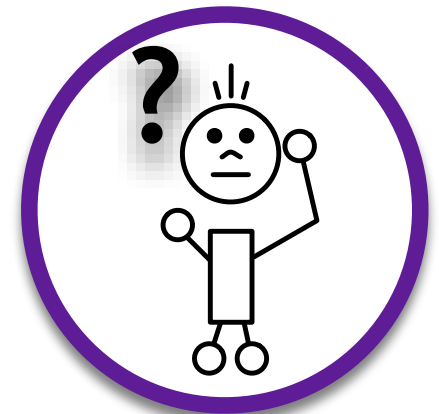
When should it vary?

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

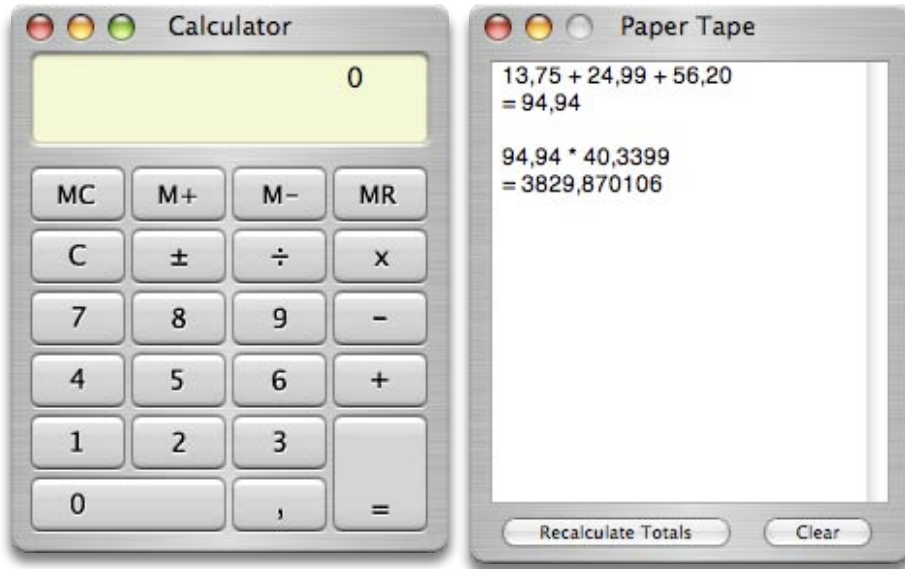
Who should make it vary?

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

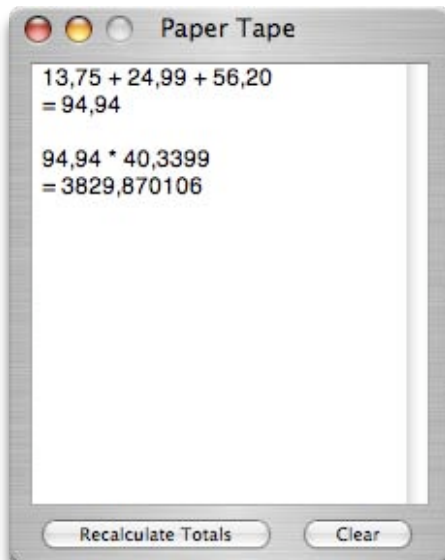
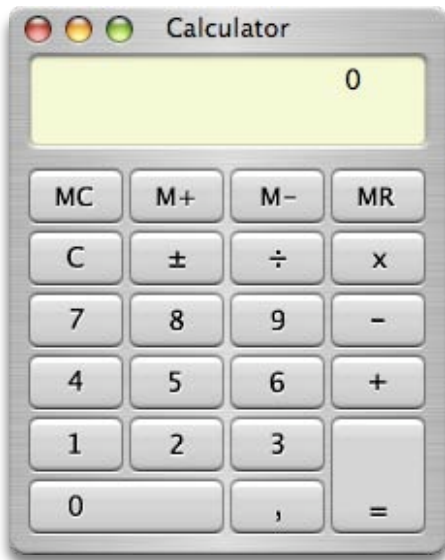
...



A Variability Example...



A Variability Example...



**What?
When?
Who?**

...



And how to do so Efficiently?

‘Efficiently’ = ‘Clean’ + ‘Easy’ + ‘Digestible’ ...

And how to do so Efficiently?

‘Efficiently’ = ‘Clean’ + ‘Easy’ + ‘Digestible’ ...

Lightweight Programmer Techniques

- Use of standard OO techniques
e.g. inheritance, polymorphism, delegation,
meta programming
- Use of design patterns
e.g. Abstract Factory, Strategy, ...
- Use of analysis patterns
e.g. Party, Portfolio, Quantity, Observation, ...



And how to do so Efficiently?

‘Efficiently’ = ‘Clean’ + ‘Easy’ + ‘Digestible’ ...

Lightweight Programmer Techniques

- Use of standard OO techniques
e.g. inheritance, polymorphism, delegation,
meta programming
- Use of design patterns
e.g. Abstract Factory, Strategy, ...
- Use of analysis patterns
e.g. Party, Portfolio, Quantity, Observation, ...



- >> Programmer sets up own “meta” infrastructure
- >> Generic code results in implicit domain knowledge
- >> Connection of domain knowledge and code is lost

C3-CoBro in a Nutshell

‘Efficiently’ = ‘Clean’ + ‘Easy’ + ‘Digestible’ ...

C3-CoBro in a Nutshell

‘Efficiently’ = ‘Clean’ + ‘Easy’ + ‘Digestible’ ...



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

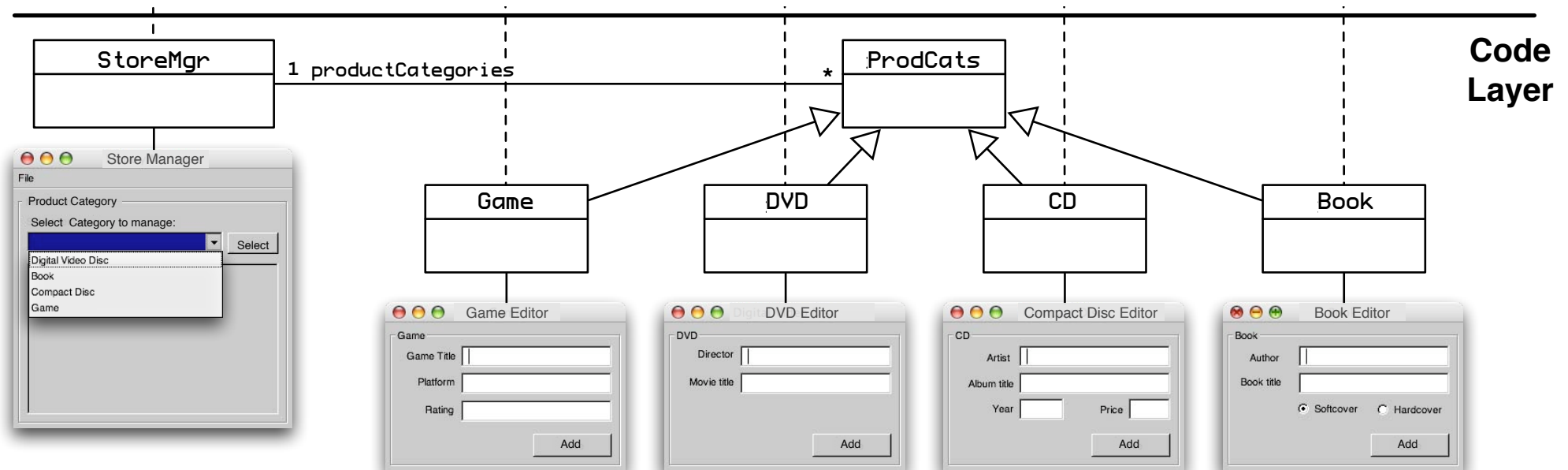
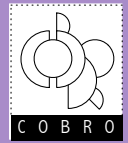
ACTIVE



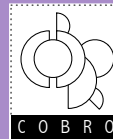
Implements the interaction with the domain knowledge environment as transparently as possible

TRANSPARENT

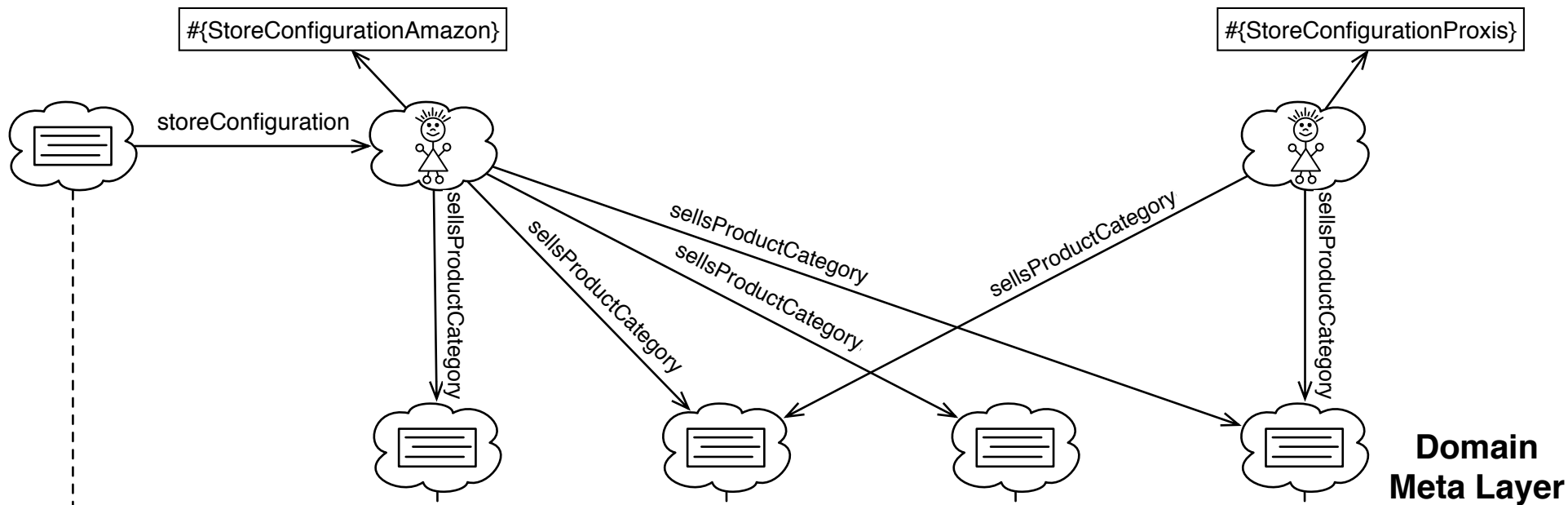
Active Domain Meta Layer



Active Domain Meta Layer

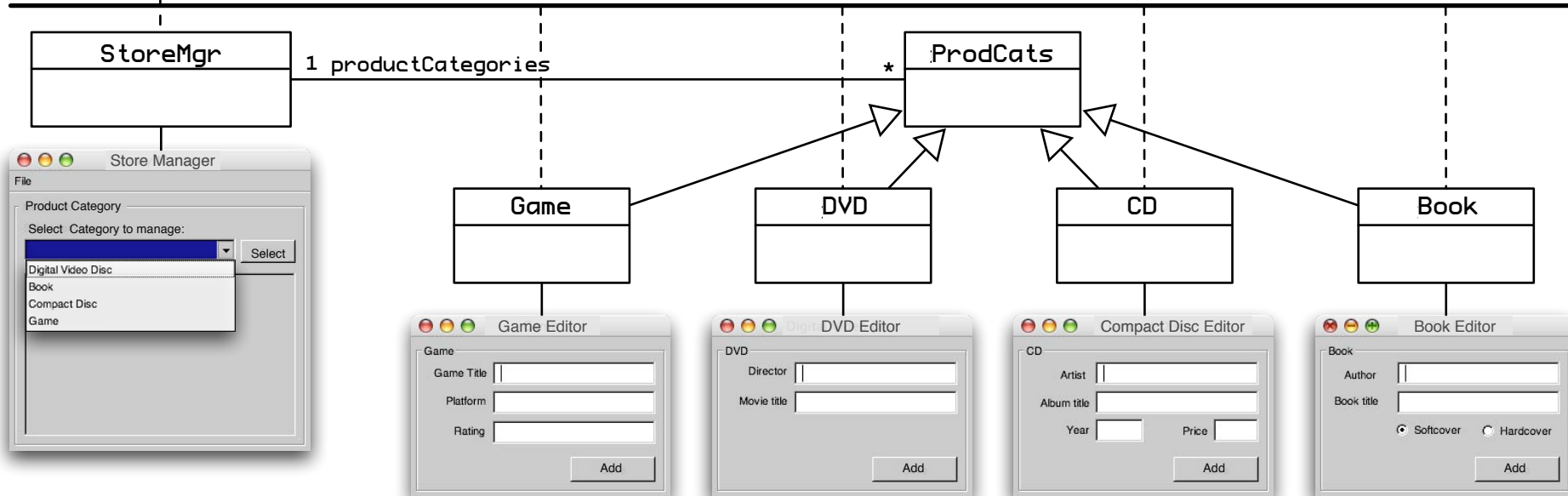


Configurational Level
(First Class Citizen)



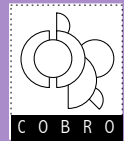
Domain
Meta Layer

Operational Level



Code
Layer

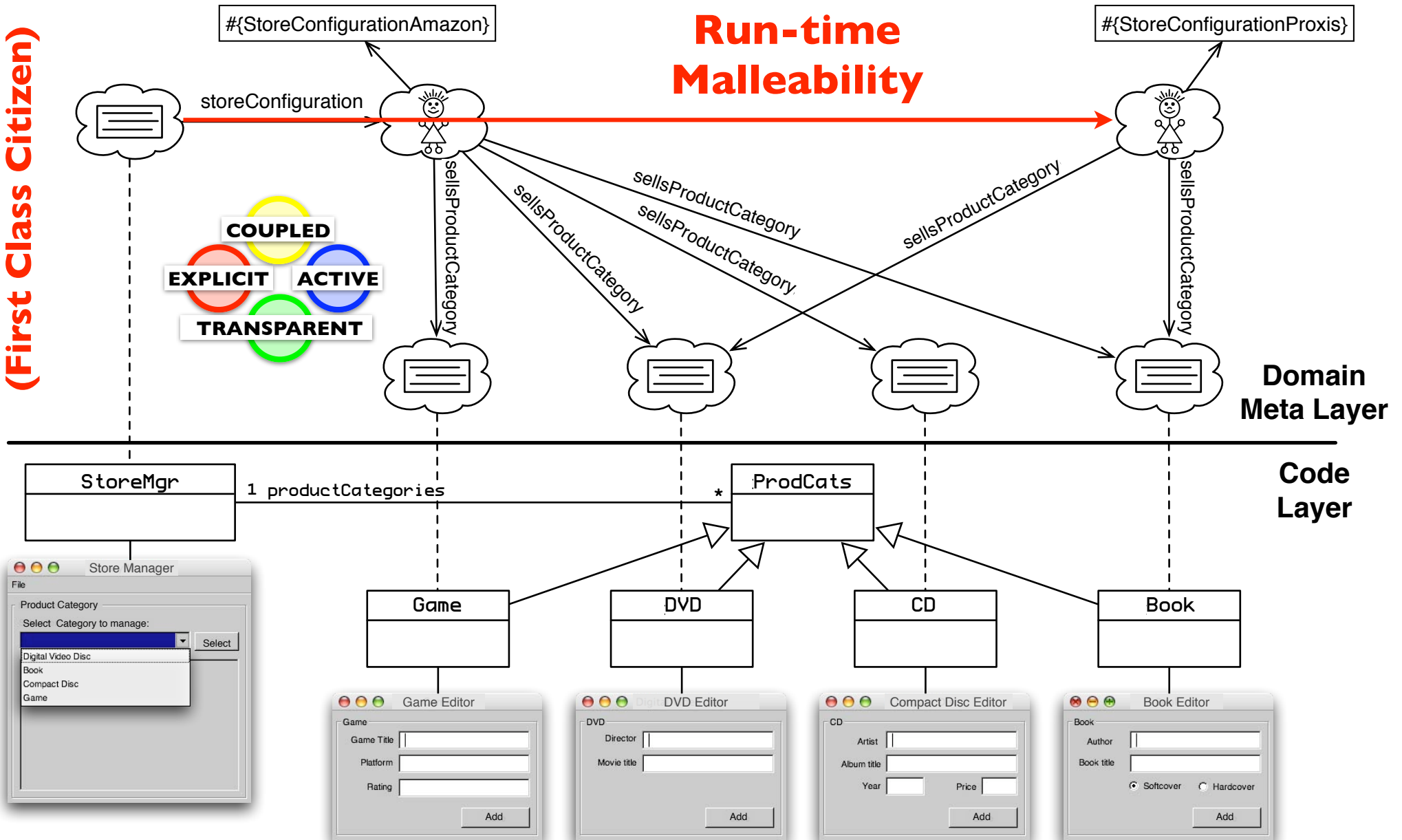
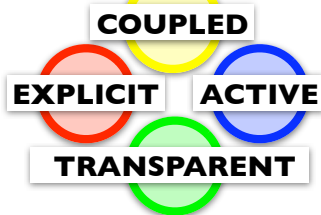
Active Domain Meta Layer



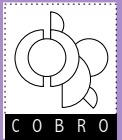
Configurational Level
(First Class Citizen)

Operational Level

Run-time Malleability



Close Integration Visualworks IDE



The screenshot displays the Visualworks IDE interface. At the top, there is a menu bar (Browser, Edit, Find, View, Package, Class, Protocol, Method, Tools, Help) and a toolbar. Below the toolbar is a package browser showing a hierarchy of packages, with 'Examples' selected. The main workspace is divided into two panes. The upper pane shows a class hierarchy diagram with the following structure:

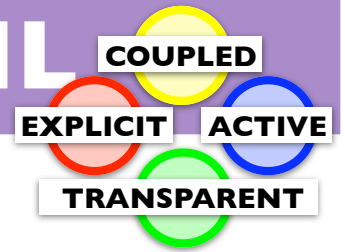
- Root.Smalltalk.StoreMgr (Class) -- storeConfiguration --> StoreConfigurationAm... (Class)
- StoreConfigurationAm... (Class) -- superconcept --> StoreConfiguration (Class)
- StoreConfigurationAm... (Class) -- superconcept --> StoreConfigurationPro... (Class)
- StoreConfigurationAm... (Class) -- sellsProductCategory --> Root.Smalltalk.DVD (Class)
- StoreConfigurationAm... (Class) -- sellsProductCategory --> Root.Smalltalk.CD (Class)
- StoreConfigurationPro... (Class) -- sellsProductCategory --> Root.Smalltalk.CD (Class)
- StoreConfigurationPro... (Class) -- sellsProductCategory --> Root.Smalltalk.Book (Class)
- StoreMgr (Class) -- stWindowSpecVisual --> Store Manager (Tool)
- StoreMgr (Class) -- stWindowSpecVisual --> DVD Editor (Tool)
- StoreMgr (Class) -- stWindowSpecVisual --> CD Editor (Tool)
- StoreMgr (Class) -- stWindowSpecVisual --> Book Editor (Tool)

The lower pane shows three tool windows: 'Store Manager' with a 'Product Category' dropdown set to 'Digital Video Disc' and an 'Add' button; 'DVD Editor' with fields for 'Director' and 'Movie Title' and an 'Add' button; 'CD Editor' with fields for 'Artist', 'Album Title', 'Year', and 'Price' and an 'Add' button; and 'Book Editor' with fields for 'Author' and 'Book Title', radio buttons for 'Softcover' and 'Hardcover', and an 'Add' button.

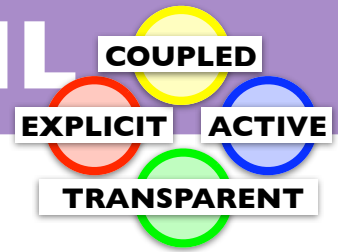
**Domain
Meta Layer**



StoreConfiguration in CoBro-CML

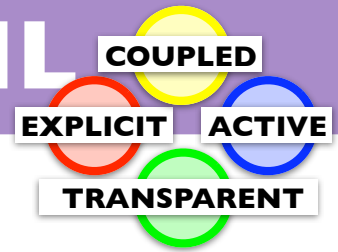


StoreConfiguration in CoBro-CML



```
(Concepts
  defineConcept: #{StoreConfigurationAmazon}
  displayName: 'StoreConfigurationAmazon'
  superconcept: Concepts.StoreConfiguration)
  sellsProductCategory: Concepts.DVD ;
  sellsProductCategory: Concepts.Book ;
  sellsProductCategory: Concepts.Game ;
  sellsProductCategory: Concepts.CD.
```

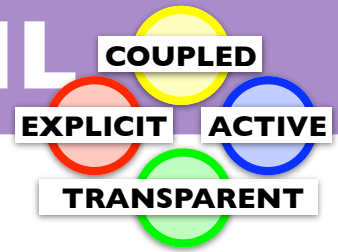
StoreConfiguration in CoBro-CML



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

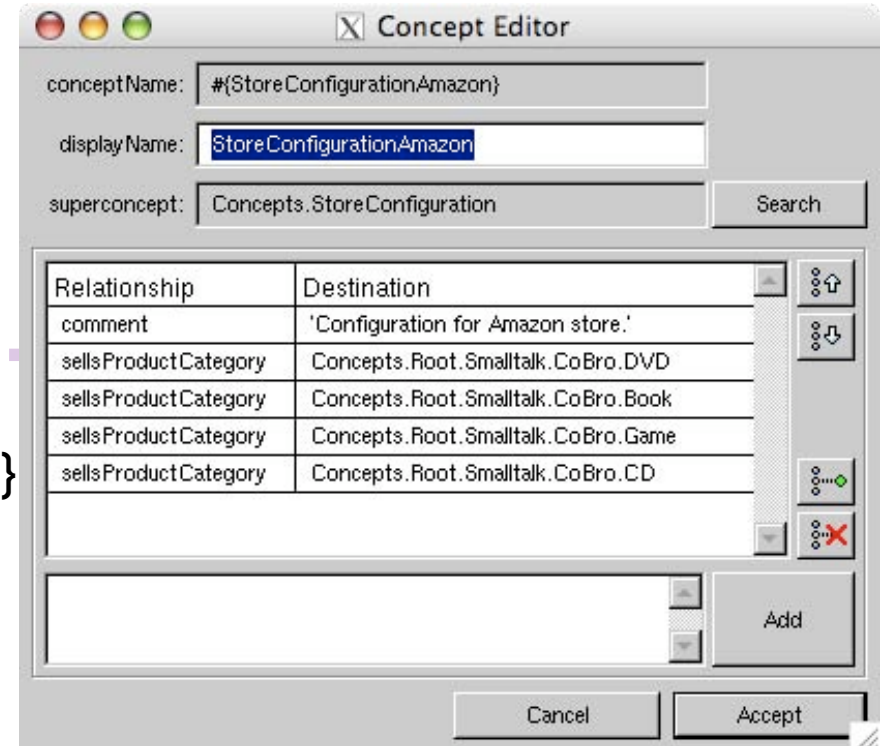
```
{Concepts
  defineConcept: #{StoreConfigurationAmazon}
  displayName: 'StoreConfigurationAmazon'
  superconcept:Concepts.StoreConfiguration)
sellsProductCategory: Concepts.DVD ;
sellsProductCategory: Concepts.Book ;
sellsProductCategory: Concepts.Game ;
sellsProductCategory: Concepts.CD.
```

StoreConfiguration in CoBro-CML

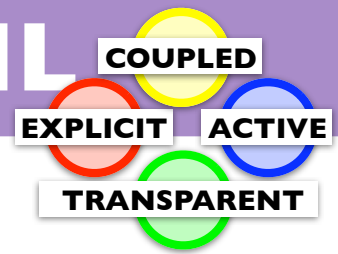


```
(Concepts
  defineConcept: #{sellsProductCategory}
  displayName: 'sellsProductCategory'
  superconcept: Concepts.DomainRelationship)
multiplicity: #(1 5) ;
destinationType: Concepts.STClass.
```

```
(Concepts
  defineConcept: #{StoreConfigurationAmazon}
  displayName: 'StoreConfigurationAmazon'
  superconcept: Concepts.StoreConfiguration)
sellsProductCategory: Concepts.DVD ;
sellsProductCategory: Concepts.Book ;
sellsProductCategory: Concepts.Game ;
sellsProductCategory: Concepts.CD.
```

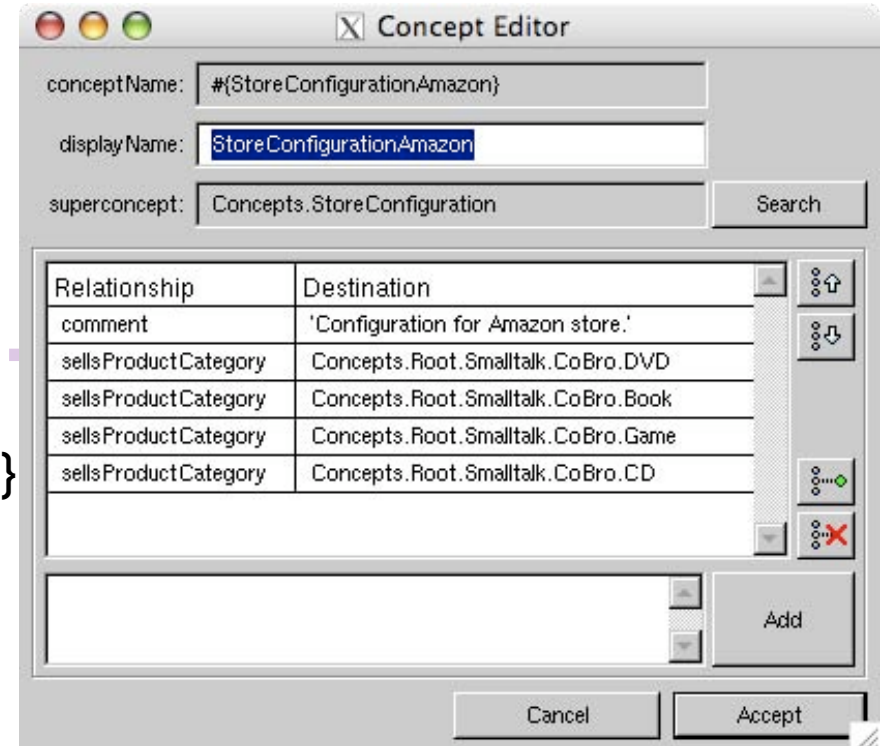


StoreConfiguration in CoBro-CML



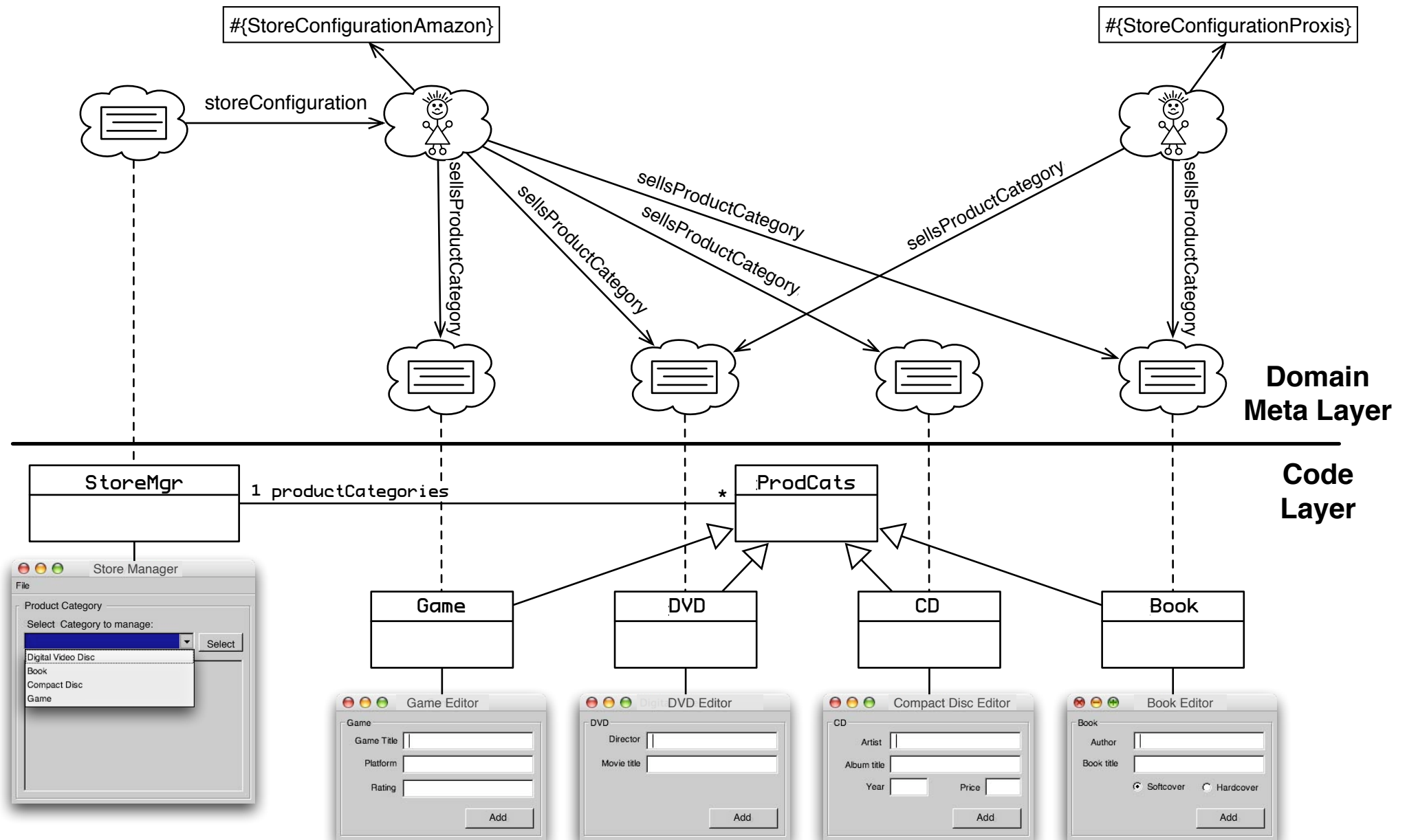
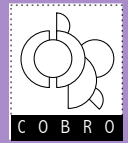
```
(Concepts
  defineConcept: #{sellsProductCategory}
  displayName: 'sellsProductCategory'
  superconcept: Concepts.DomainRelationship)
multiplicity: #(1 5) ;
destinationType: Concepts.STClass.
```

```
(Concepts
  defineConcept: #{StoreConfigurationAmazon}
  displayName: 'StoreConfigurationAmazon'
  superconcept: Concepts.StoreConfiguration)
sellsProductCategory: Concepts.DVD ;
sellsProductCategory: Concepts.Book ;
sellsProductCategory: Concepts.Game ;
sellsProductCategory: Concepts.CD.
```

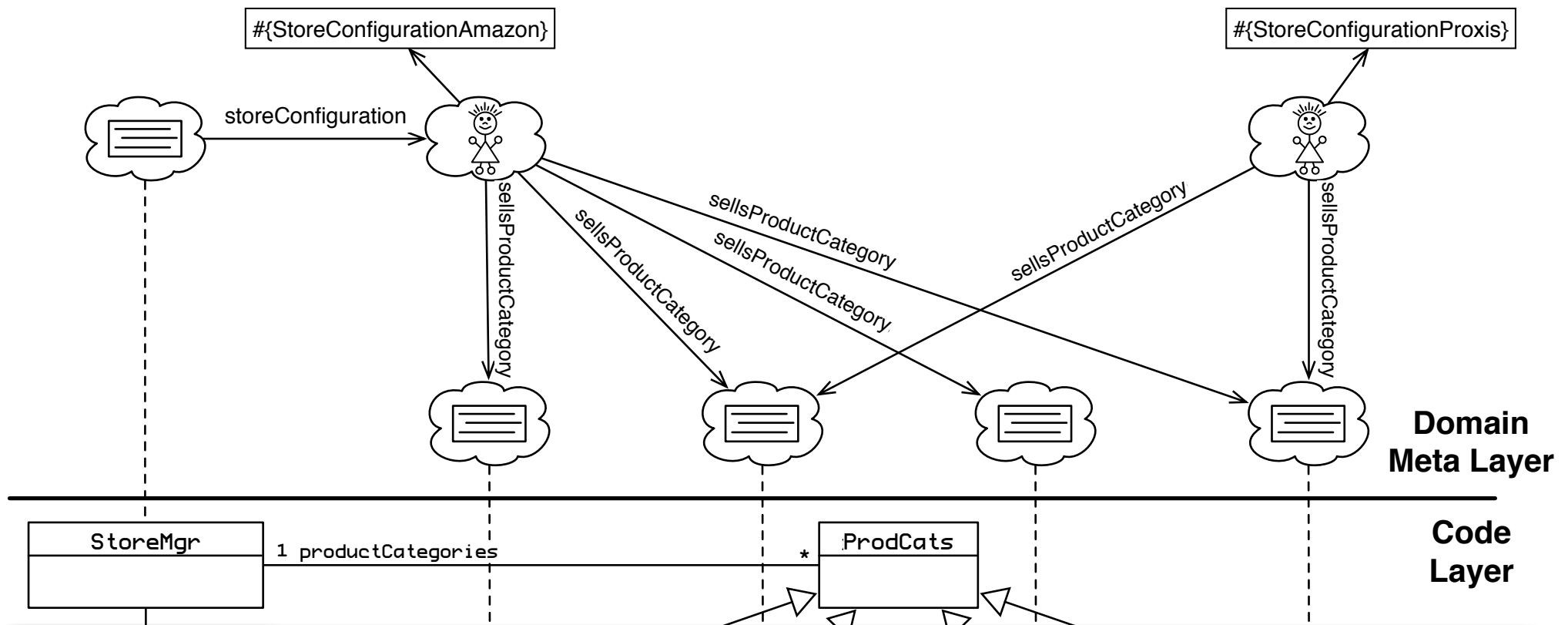
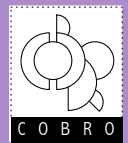


```
Concepts.StoreMgr storeConfiguration: StoreConfigurationAmazon.
```

Mixing CoBro-CML with Smalltalk



Mixing CoBro-CML with Smalltalk



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

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**

...

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**

...

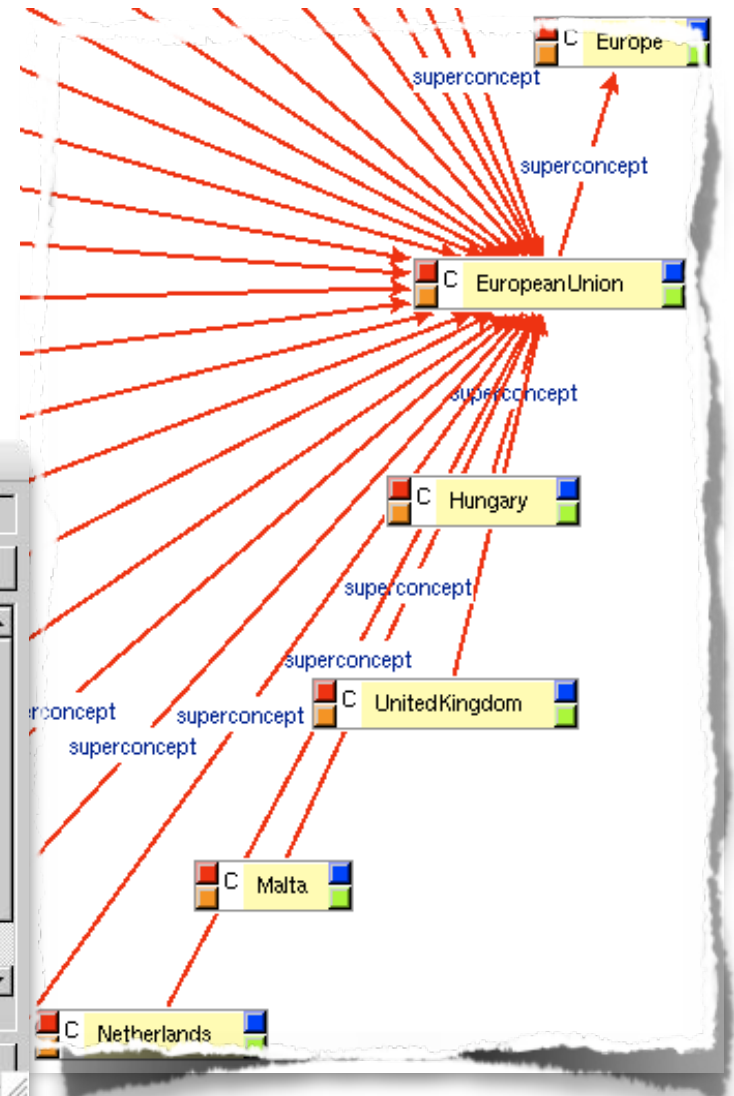
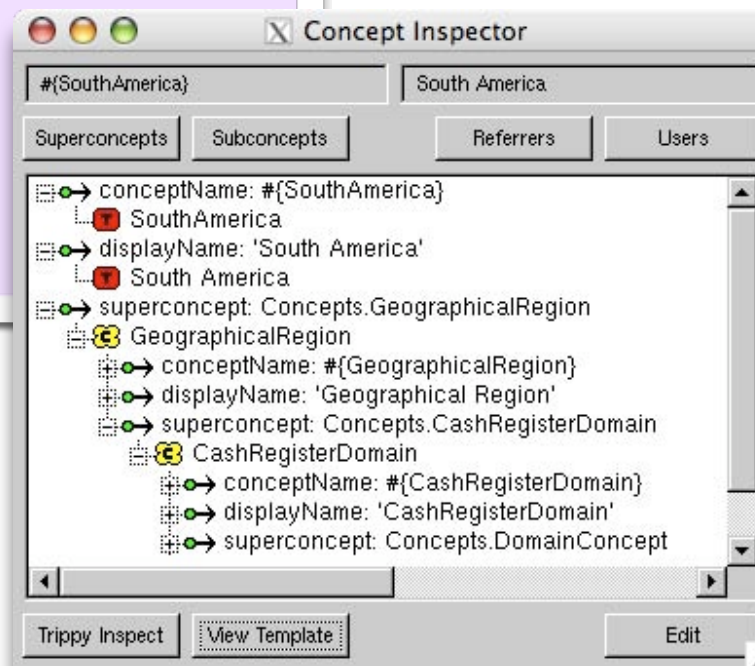
(Concepts

```
defineConcept: #{SouthAmerica}
```

```
displayName: 'South America'
```

```
superconcept: Concepts.GeographicalRegion).
```

...



More Variability: ShippingRestrictions

(Concepts

```
defineConcept: #{RestrictionForInternational}
displayName: 'Shipping Restriction For International'
superconcept: Concepts.ShippingRestriction)
restrictedLocations: #{Concepts.LatinAmerica Concepts.Israel
                      Concepts.Africa ...} ;
restrictedProductCategories: #{Concepts.Jewelry Concepts.KitchenItems
                               Concepts.SportingGoods ...} ;
comment: 'Items of our market place can be ...'.
```

...

More Variability: ShippingRestrictions

(Concepts

```
defineConcept: #{RestrictionForInternational}
displayName: 'Shipping Restriction For International'
superconcept: Concepts.ShippingRestriction)
restrictedLocations: #{Concepts.LatinAmerica Concepts.Israel
                      Concepts.Africa ...} ;
restrictedProductCategories: #{Concepts.Jewelry Concepts.KitchenItems
                               Concepts.SportingGoods ...} ;
comment: 'Items of our market place can be ...'.
```

U.S. Protectorates (including Puerto Rico)

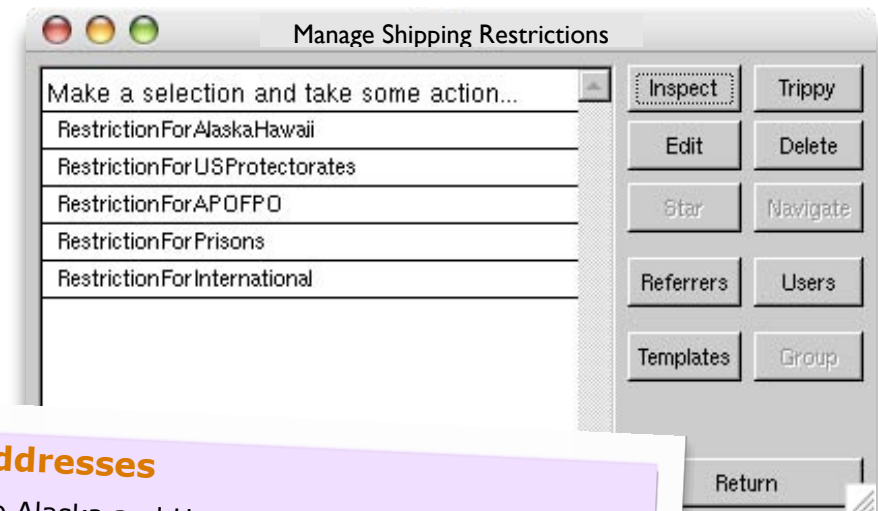
Only the following items **can** be shipped to U.S. Protectorates:

- 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

Fun with Validators & Interpreters

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

Fun with Validators & Interpreters

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

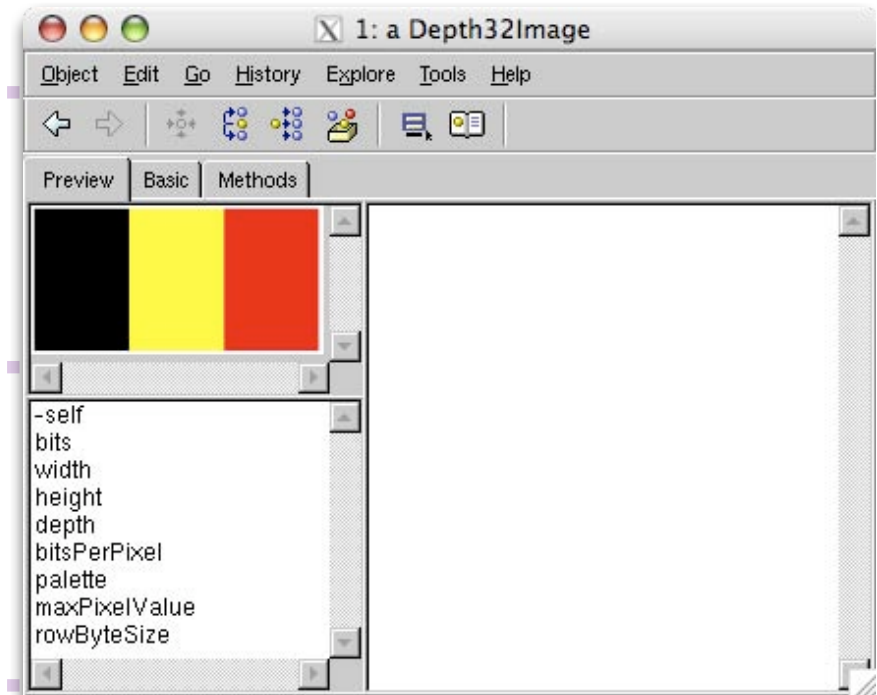
```
{Concepts
  defineConcept: #{Belgium}
  displayName: 'Belgium'
  superconcept: Concepts.EuropeanUnion)
  flag: 'Flags/Flag_of_Belgium.png'.
```

Fun with Validators & Interpreters

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

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

Concepts.Belgium flag inspect



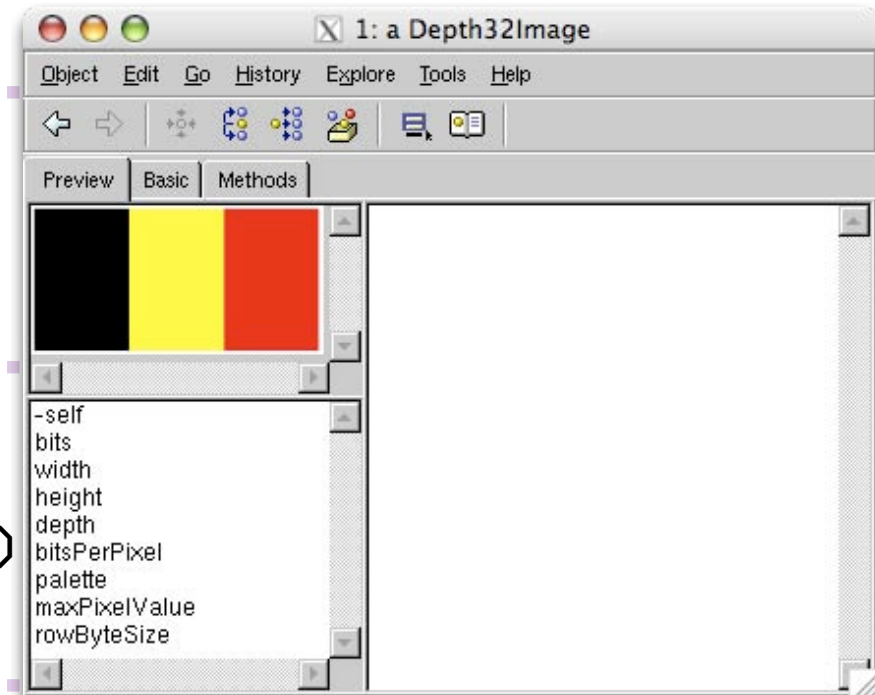
Fun with Validators & Interpreters

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

```
(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.Belgium flag inspect



Fun with Validators & Interpreters

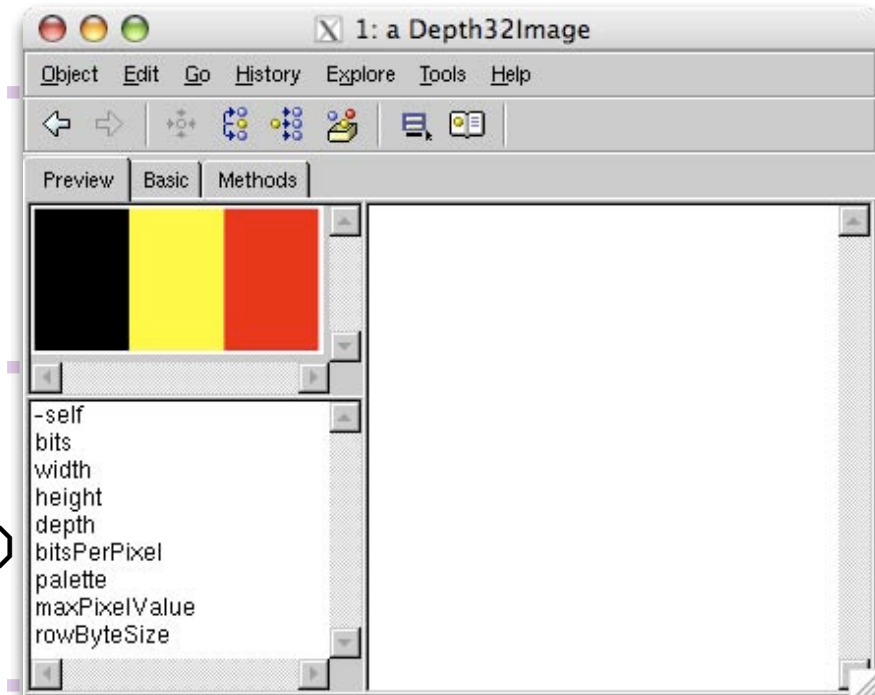
```
Concepts.StoreConfiguration
configurationValidator:
  '[:config |
    config sellsProductCategory reject:
      [:each | each asSmalltalk
        inheritsFrom: ProdCat]]'
```

```
(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]'
```

Concepts.Belgium flag inspect



Future Work

Future Work

- Feature Diagrams to model the variability

Future Work

- Feature Diagrams to model the variability
- Explore the power of valueValidators & valueInterpreters

Future Work

- Feature Diagrams to model the variability
- Explore the power of valueValidators & valueInterpreters
- Explore CoBro as a Meta-CASE tool

Future Work

- Feature Diagrams to model the variability
- Explore the power of valueValidators & valueInterpreters
- Explore CoBro as a Meta-CASE tool
- Establish “Methodology”-side of C3

Future Work

- Feature Diagrams to model the variability
- Explore the power of valueValidators & valueInterpreters
- Explore CoBro as a Meta-CASE tool
- Establish “Methodology”-side of C3
 - Guidelines, Patterns, Considerations, ...

Future Work

- Feature Diagrams to model the variability
- Explore the power of valueValidators & valueInterpreters
- Explore CoBro as a Meta-CASE tool
- Establish “Methodology”-side of C3
 - Guidelines, Patterns, Considerations, ...
- How to check if the variability is indeed supported “efficiently”? Quality Measures

W. Hogarth - "The Analysis of Beauty"



Quality Measures?

- Aesthetic
- Concise
- Self-descriptive
- Extensible ...

Conclusion

Conclusion

Each technology for variability has its particular advantages/drawbacks

Conclusion

Each technology for variability has its particular advantages/drawbacks

Always consider the cost/benefit of variant features!

Conclusion

Each technology for variability has its particular advantages/drawbacks

Always consider the cost/benefit of variant features!

Introducing software variability requires the correct state of mind for all perspectives

Conclusion

Each technology for variability has its particular advantages/drawbacks

Always consider the cost/benefit of variant features!

Introducing software variability requires the correct state of mind for all perspectives

**Renewed interest in dynamic languages
a coincidence?**

Questions?



Vrije Universiteit Brussel

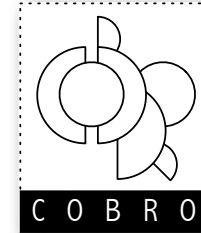
Faculty of Sciences

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/>