RELEASeD

REsearch Lab on software Evolution And software Development technology

Prof. Kim Mens

Software Evolution	**Co-evolving** program code & design
	Reverse engineering & program understanding
	Program **querying**
	Source code **mining** and aspect mining
	Enforcing program & design **regularities**
	Diagnosing & resolving detected inconsistencies
Software Development	**Language** engineering (AOP, COP, AmI)
	Program **transformation**
	Generative programming
	Declarative metaprogramming

Dr. Johan Brichau

Sebastián González

Diego Ordóñez

Sergio Castro

Alfredo Cádiz

Programming technology, languages, formalisms and tools
for **software development, maintenance and evolution**

Mining Source Code for design regularities (work in progress)

Kim Mens
RELEASeD group
Département d’ingénierie Informatique
Université catholique de Louvain

Andy Kellens
Programming Technology Lab
Vakgroep Informatica
Vrije Universiteit Brussel

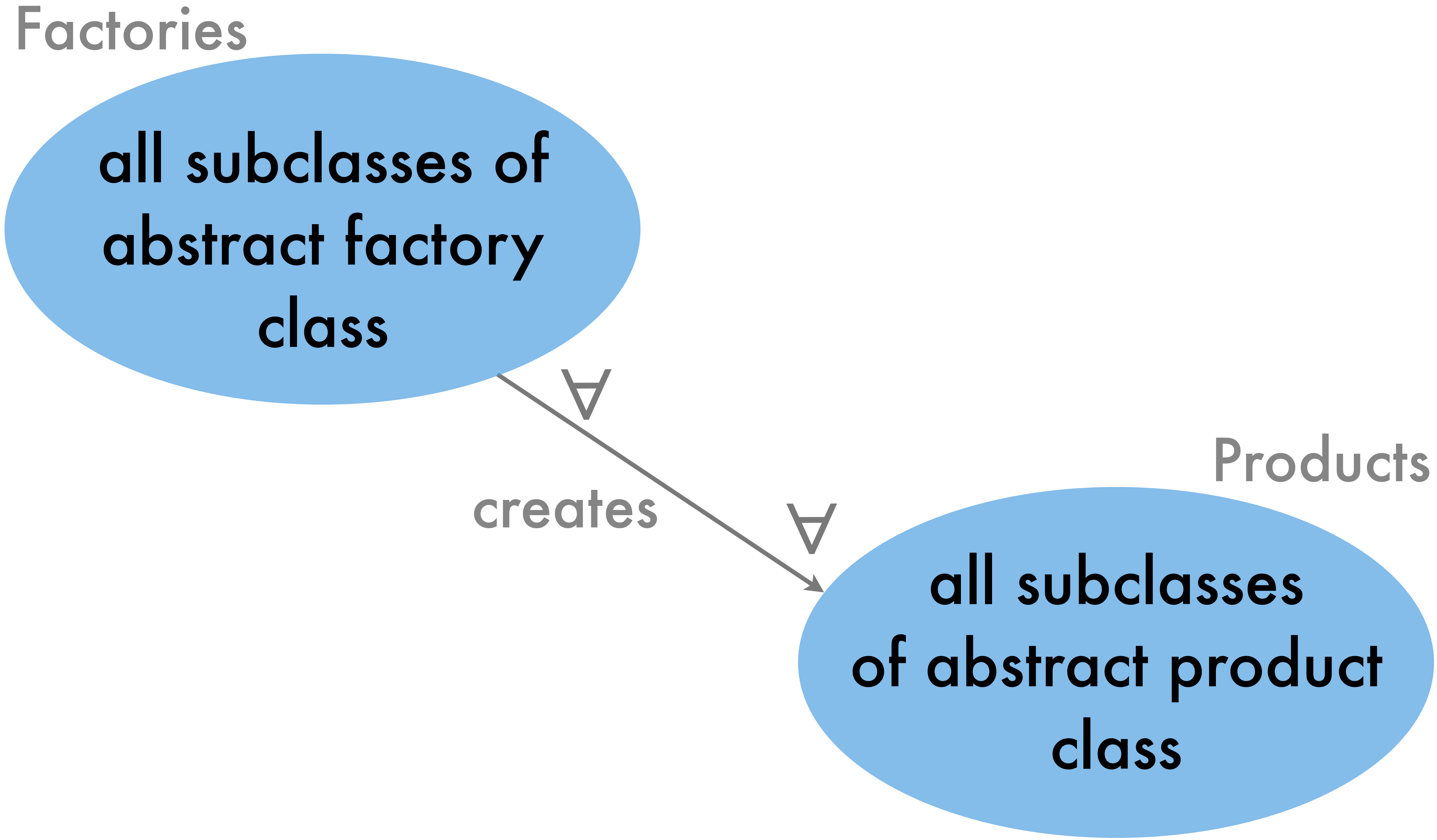
Mining Source Code for design regularities (work in progress)

Kim Mens
RELEASeD group
Département d'ingénierie Informatique
Université catholique de Louvain

Andy Kellens
Programming Technology Lab
Vakgroep Informatica
Vrije Universiteit Brussel

(Gabriela Arevalo, La Plata)

Context: Intensional Views



Factories

all subclasses of abstract factory class

∀

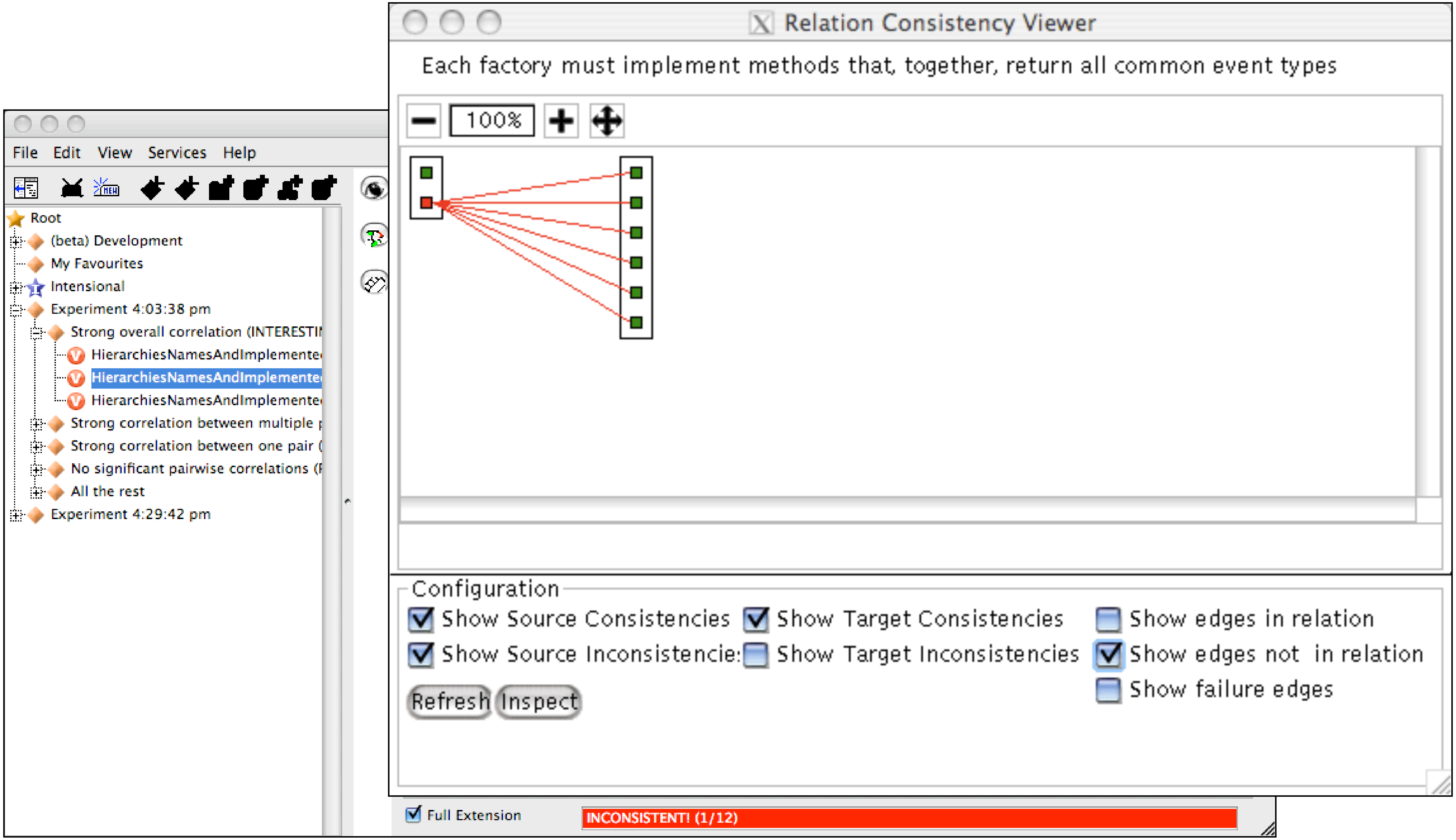
creates

∀

Products

all subclasses of abstract product class

Context: Intensional Views



Root
(beta) Development
My Favourites
Intensional
Experiment 4:03:38 pm
Strong overall correlation (INTERESTI
HierarchiesNamesAndImplemente
HierarchiesNamesAndImplemente
HierarchiesNamesAndImplemente
Strong correlation between multiple
Strong correlation between one pair (
No significant pairwise correlations (F
All the rest
Experiment 4:29:42 pm

Relation Consistency Viewer

Each factory must implement methods that, together, return all common event types

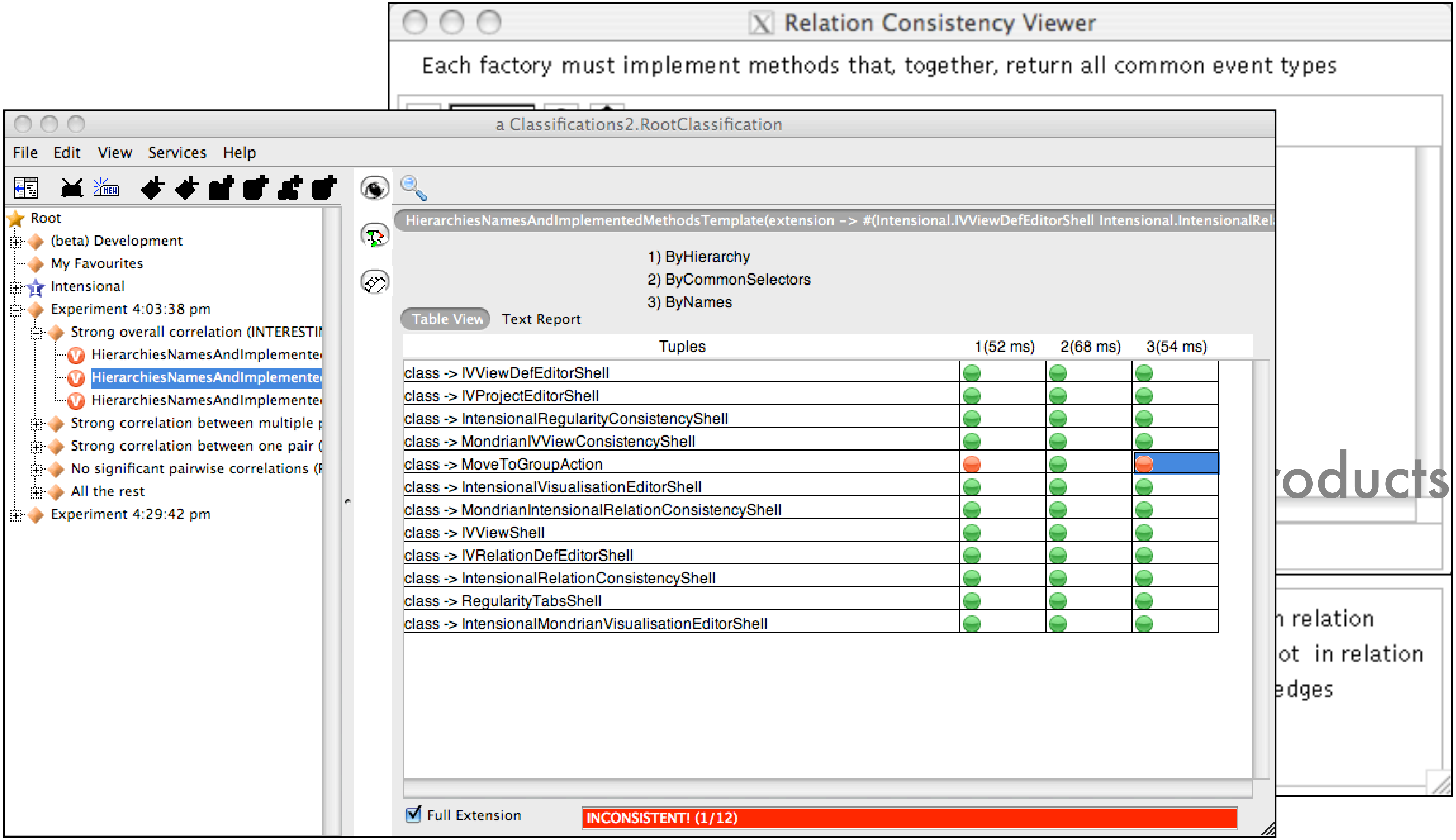
Configuration

☑ Show Source Consistencies ☑ Show Target Consistencies □ Show edges in relation
☑ Show Source Inconsistencie: □ Show Target Inconsistencies ☑ Show edges not in relation
□ Show failure edges

☑ Full Extension

INCONSISTENT! (1/12)

Context: Intensional Views



Relation Consistency Viewer

Each factory must implement methods that, together, return all common event types

roducts

n relation

ot in relation

edges

a Classifications2.RootClassification

- Root
  - (beta) Development
  - My Favourites
  - Intensional
  - Experiment 4:03:38 pm
    - Strong overall correlation (INTERESTI
      - HierarchiesNamesAndImplemente
      - HierarchiesNamesAndImplemente
      - HierarchiesNamesAndImplemente
    - Strong correlation between multiple
    - Strong correlation between one pair (
    - No significant pairwise correlations (
    - All the rest
  - Experiment 4:29:42 pm

HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.IVViewDefEditorShell Intensional.IntensionalRel

1) ByHierarchy
2) ByCommonSelectors
3) ByNames

Tuples	1(52 ms)	2(68 ms)	3(54 ms)
class -> IVViewDefEditorShell	🟢	🟢	🟢
class -> IVProjectEditorShell	🟢	🟢	🟢
class -> IntensionalRegularityConsistencyShell	🟢	🟢	🟢
class -> MondrianIVViewConsistencyShell	🟢	🟢	🟢
class -> MoveToGroupAction	🔴	🟢	🔴
class -> IntensionalVisualisationEditorShell	🟢	🟢	🟢
class -> MondrianIntensionalRelationConsistencyShell	🟢	🟢	🟢
class -> IVViewShell	🟢	🟢	🟢
class -> IVRelationDefEditorShell	🟢	🟢	🟢
class -> IntensionalRelationConsistencyShell	🟢	🟢	🟢
class -> RegularityTabsShell	🟢	🟢	🟢
class -> IntensionalMondrianVisualisationEditorShell	🟢	🟢	🟢

☑ Full Extension

INCONSISTENT! (1/12)

Motivation: Documenting and Verifying Design Regularities

- Context: Intensional views
  - Document & verify design regularities in source code
  - Hard to document all regularities “by hand”
- Need for automation
  - program comprehension techniques to discover these views and regularities
  - Source code mining techniques in particular

Challenge: extracting design regularities from source code

- How to extract views & regularities from code?
- Need for automated code mining techniques
  - Similar to aspect mining
  - Based on data mining / code analysis / program understanding techniques
  - FCA, clustering, clone detection, program slicing ...
- Extra difficulty: extract *intension*, not only extension
  - not only the elements but also *why* they are related

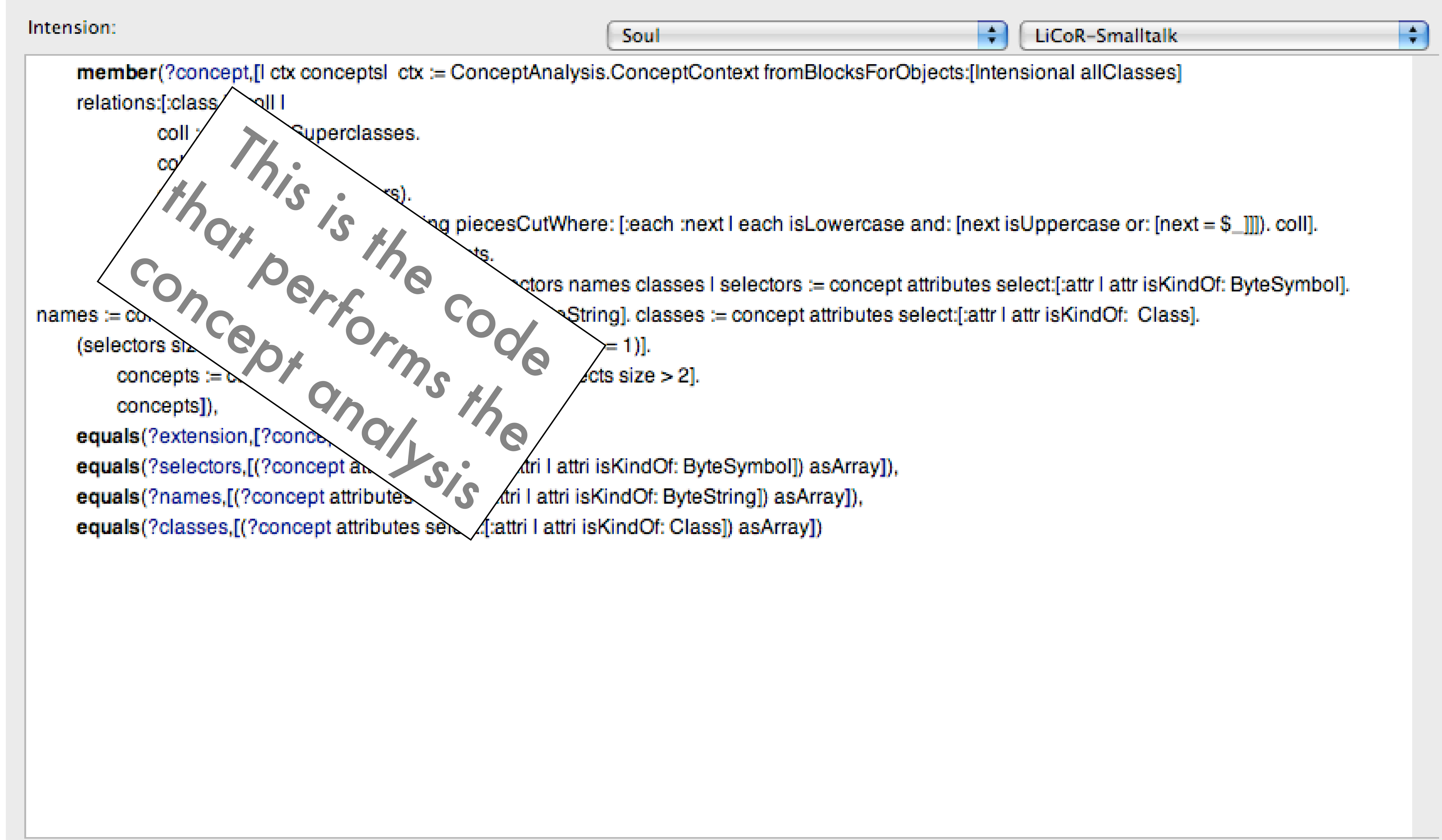
Ongoing experiment

- A clever combination of
  - Formal concept analysis (FCA)
  - Intensional views
    - alternative views
    - parameterized views (to generate views from a template description)
  - Automated classification and filtering
  - Manual analysis, validation and refinement of results

Ongoing experiment: step 1

- Step 1: Formal concept analysis
  - (finds groups of “objects” with shared “properties”)
  - objects: all classes
  - attributes: (we mix 3 kinds of attributes in 1 analysis)
    - have methods with similar name (keyword shared)
    - implement same method (selector shared)
    - in same hierarchy (parent classes shared)

Ongoing experiment: step 1 (details)



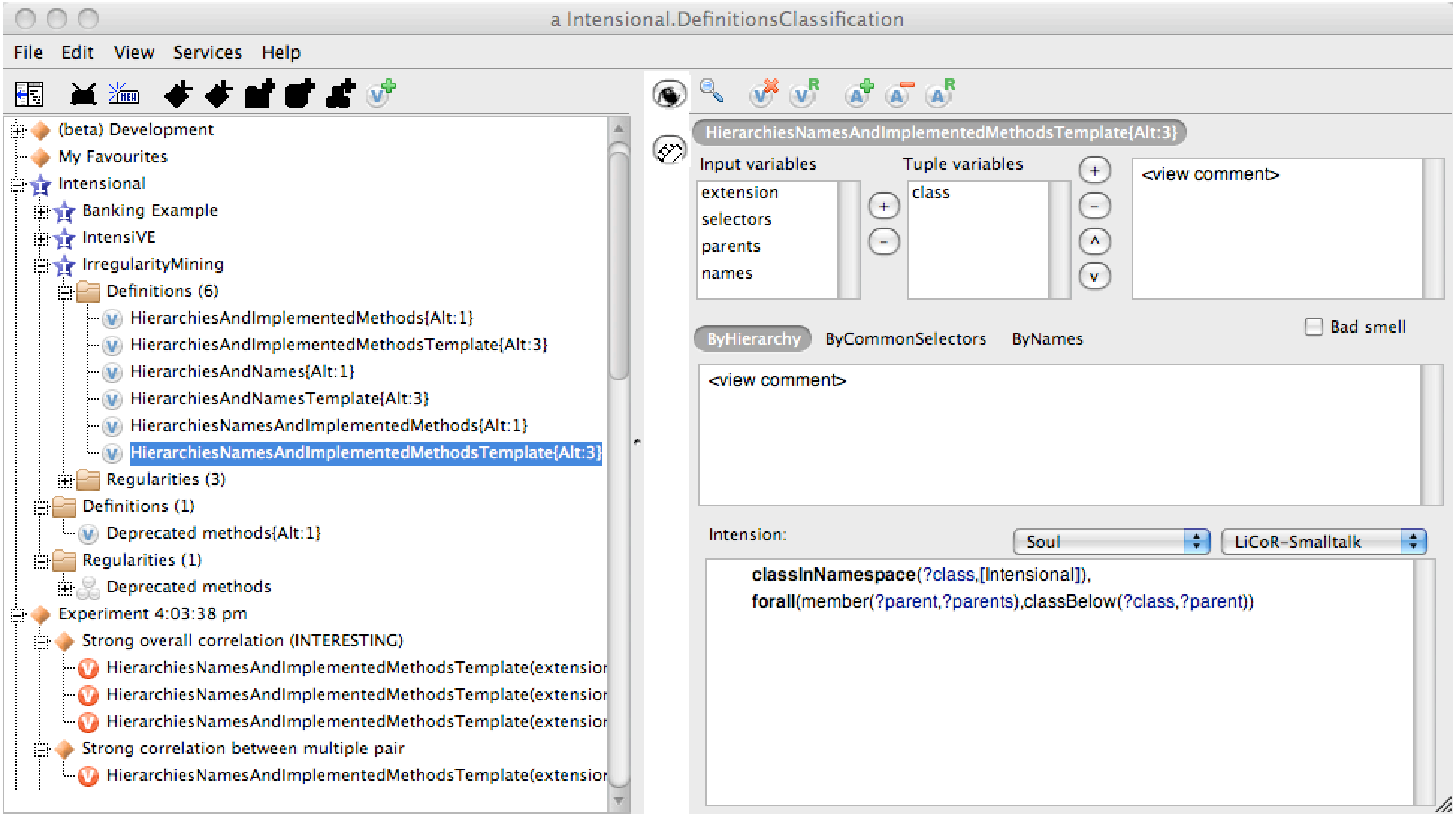
```
	member(?concept,[| ctx concepts|  ctx := ConceptAnalysis.ConceptContext fromBlocksForObjects:[Intensional allClasses]
	relations:[:class | coll |
		coll :              Superclasses.
		co
		                      rs).
		                      ng piecesCutWhere: [:each :next | each isLowercase and: [next isUppercase or: [next = $_]]]). coll].
		                        ts.
		                        ctors names classes | selectors := concept attributes select:[:attr | attr isKindOf: ByteSymbol].
names := co                     String]. classes := concept attributes select:[:attr | attr isKindOf:  Class].
	(selectors si                              = 1)].
		concepts := c                          cts size > 2].
		concepts]),
	equals(?extension,[?conce
	equals(?selectors,[(?concept at              ttri | attri isKindOf: ByteSymbol]) asArray]),
	equals(?names,[(?concept attributes          ttri | attri isKindOf: ByteString]) asArray]),
	equals(?classes,[(?concept attributes sel     :[:attri | attri isKindOf: Class]) asArray])
```

This is the code that performs the concept analysis

Ongoing experiment: step 2

- Result step 1 = many potentially interesting “concepts”
  - groups of classes that may share similar names, selectors and parent classes
- Step 2: generate an intensional view for each of these
  - Elements of the view = all the classes in the group
  - Alternative 1 : all classes with shared keywords
  - Alternative 2 : all classes with shared names
  - Alternative 3 : all classes with shared parents

Ongoing experiment: step 2 (details) Parameterized view with 3 alternatives



- (beta) Development
- My Favourites
- Intensional
  - Banking Example
  - IntensiVE
  - IrregularityMining
    - Definitions (6)
      - HierarchiesAndImplementedMethods{Alt:1}
      - HierarchiesAndImplementedMethodsTemplate{Alt:3}
      - HierarchiesAndNames{Alt:1}
      - HierarchiesAndNamesTemplate{Alt:3}
      - HierarchiesNamesAndImplementedMethods{Alt:1}
      - HierarchiesNamesAndImplementedMethodsTemplate{Alt:3}
    - Regularities (3)
  - Definitions (1)
    - Deprecated methods{Alt:1}
  - Regularities (1)
    - Deprecated methods
- Experiment 4:03:38 pm
  - Strong overall correlation (INTERESTING)
    - HierarchiesNamesAndImplementedMethodsTemplate(extensio
    - HierarchiesNamesAndImplementedMethodsTemplate(extensio
    - HierarchiesNamesAndImplementedMethodsTemplate(extensio
  - Strong correlation between multiple pair
    - HierarchiesNamesAndImplementedMethodsTemplate(extensio

HierarchiesNamesAndImplementedMethodsTemplate{Alt:3}

Input variables	Tuple variables
extension	class
selectors	
parents	
names	

<view comment>

□ Bad smell

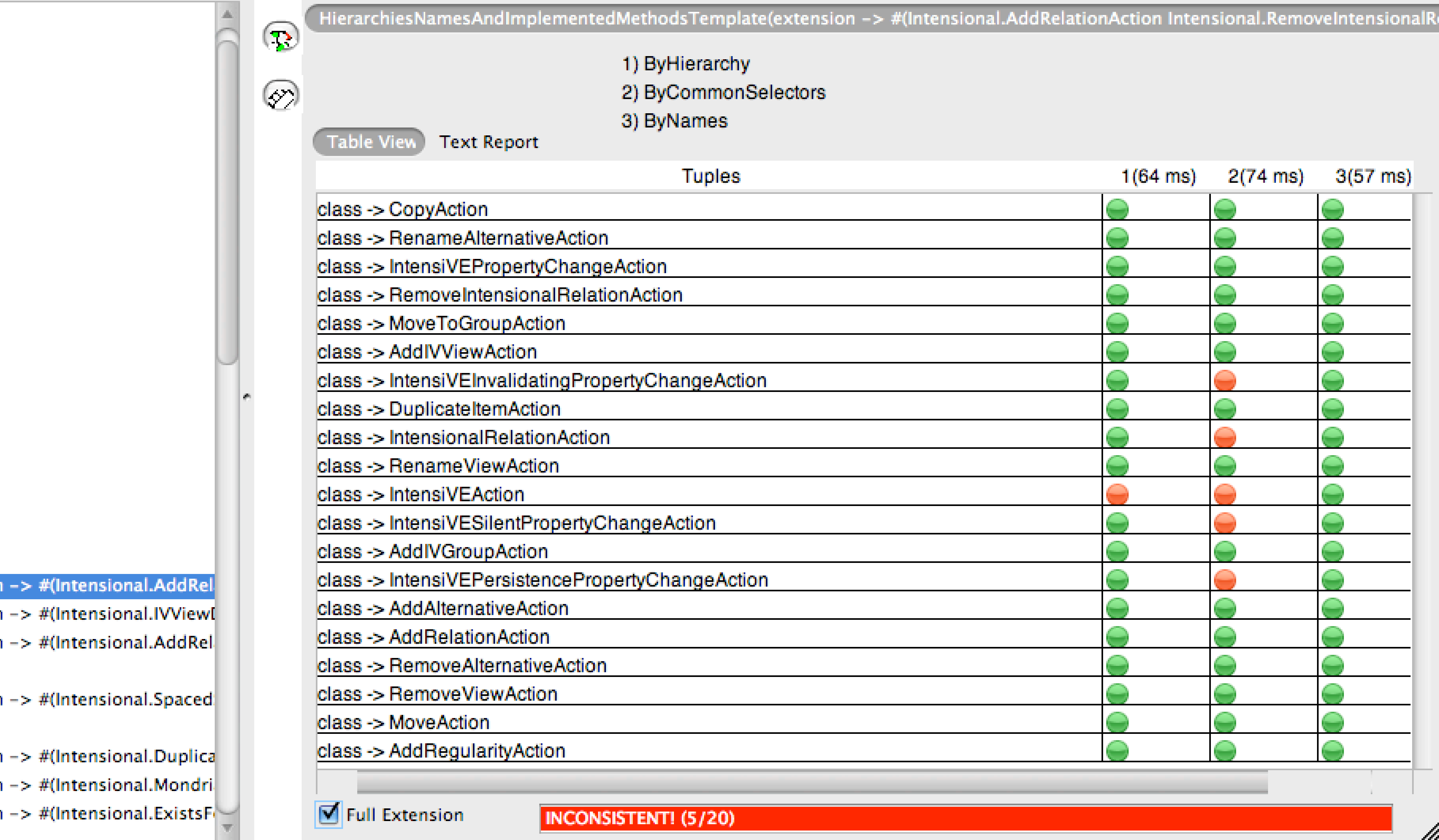
ByHierarchy ByCommonSelectors ByNames

<view comment>

Intension: Soul LiCoR-Smalltalk

```
classInNamespace(?class,[Intensional]),
forall(member(?parent,?parents),classBelow(?class,?parent))
```

Ongoing experiment: step 2 (details)
Analyze the generated views



n -> #(Intensional.AddRel
n -> #(Intensional.IVViewI
n -> #(Intensional.AddRel
n -> #(Intensional.Spaced
n -> #(Intensional.Duplica
n -> #(Intensional.Mondri
n -> #(Intensional.ExistsF

HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.AddRelationAction Intensional.RemoveIntensionalR

1) ByHierarchy
2) ByCommonSelectors
3) ByNames

Tuples	1(64 ms)	2(74 ms)	3(57 ms)
class -> CopyAction	🟢	🟢	🟢
class -> RenameAlternativeAction	🟢	🟢	🟢
class -> IntensiVEPropertyChangeAction	🟢	🟢	🟢
class -> RemoveIntensionalRelationAction	🟢	🟢	🟢
class -> MoveToGroupAction	🟢	🟢	🟢
class -> AddIVViewAction	🟢	🟢	🟢
class -> IntensiVEInvalidatingPropertyChangeAction	🟢	🔴	🟢
class -> DuplicateItemAction	🟢	🟢	🟢
class -> IntensionalRelationAction	🟢	🔴	🟢
class -> RenameViewAction	🟢	🟢	🟢
class -> IntensiVEAction	🔴	🔴	🟢
class -> IntensiVESilentPropertyChangeAction	🟢	🔴	🟢
class -> AddIVGroupAction	🟢	🟢	🟢
class -> IntensiVEPersistencePropertyChangeAction	🟢	🔴	🟢
class -> AddAlternativeAction	🟢	🟢	🟢
class -> AddRelationAction	🟢	🟢	🟢
class -> RemoveAlternativeAction	🟢	🟢	🟢
class -> RemoveViewAction	🟢	🟢	🟢
class -> MoveAction	🟢	🟢	🟢
class -> AddRegularityAction	🟢	🟢	🟢

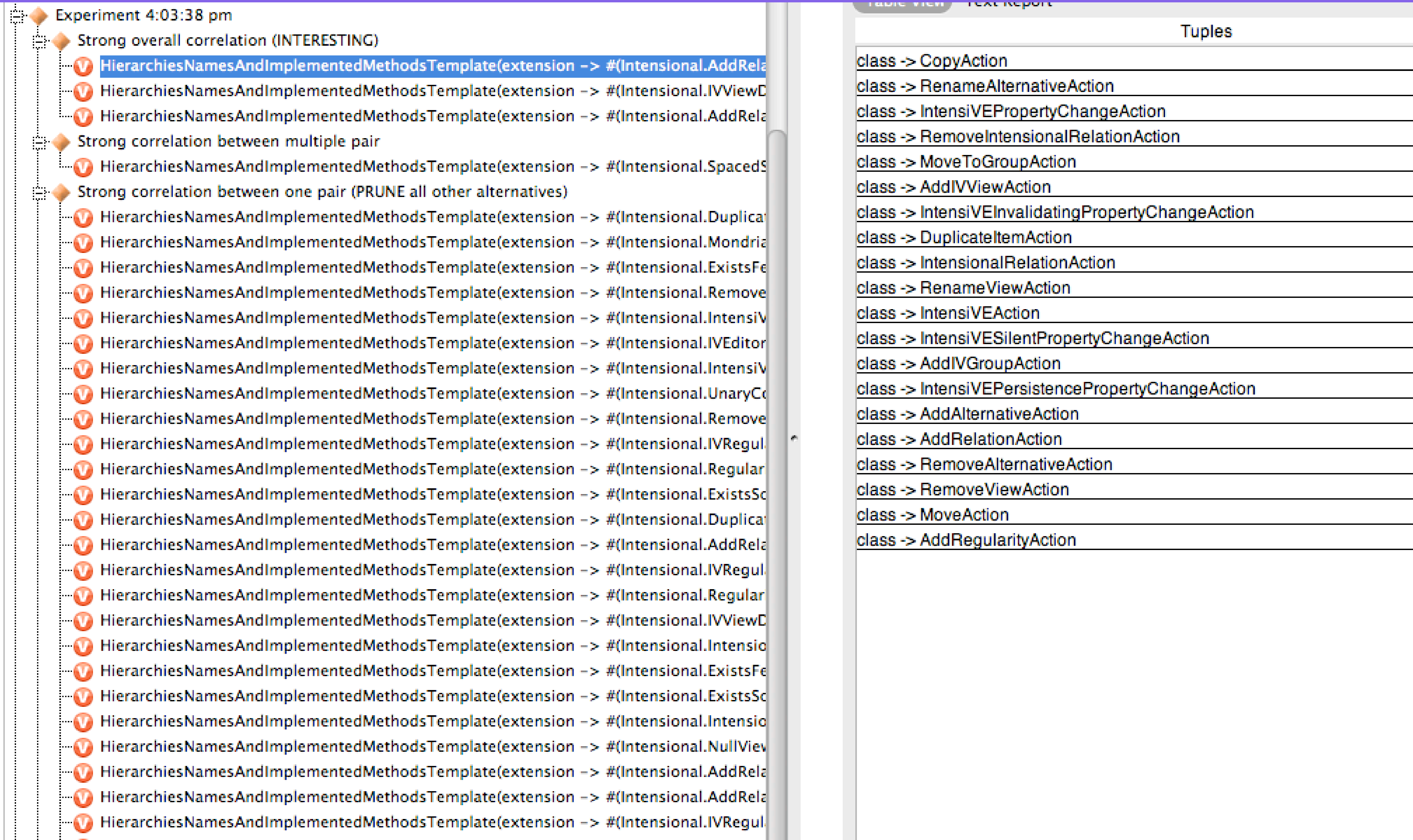
☑ Full Extension

INCONSISTENT! (5/20)

Ongoing experiment: step 3

- Result step 2 = many views
  - some with strongly correlated properties
  - others with much less correlation (less interesting)
- Step 3: automatically separate interesting views from less interesting ones based on % of correlation
- Step 4: analyse resulting views manually to confirm or discard interesting regularities in the source code

Ongoing experiment: step 3 (details) Automatic classification



- Experiment 4:03:38 pm
  - Strong overall correlation (INTERESTING)
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.AddRela
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.IVViewD
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.AddRela
  - Strong correlation between multiple pair
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.SpacedS
  - Strong correlation between one pair (PRUNE all other alternatives)
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.Duplica
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.Mondria
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.ExistsFe
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.Remove
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.IntensiV
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.IVEditor
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.IntensiV
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.UnaryCo
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.Remove
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.IVRegul
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.Regular
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.ExistsSo
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.Duplica
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.AddRela
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.IVRegul
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.Regular
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.IVViewD
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.Intensio
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.ExistsFe
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.ExistsSo
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.Intensio
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.NullView
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.AddRela
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.AddRela
    - HierarchiesNamesAndImplementedMethodsTemplate(extension -> #(Intensional.IVRegul

Tuples
class -> CopyAction
class -> RenameAlternativeAction
class -> IntensiVEPropertyChangeAction
class -> RemoveIntensionalRelationAction
class -> MoveToGroupAction
class -> AddIVViewAction
class -> IntensiVEInvalidatingPropertyChangeAction
class -> DuplicateItemAction
class -> IntensionalRelationAction
class -> RenameViewAction
class -> IntensiVEAction
class -> IntensiVESilentPropertyChangeAction
class -> AddIVGroupAction
class -> IntensiVEPersistencePropertyChangeAction
class -> AddAlternativeAction
class -> AddRelationAction
class -> RemoveAlternativeAction
class -> RemoveViewAction
class -> MoveAction
class -> AddRegularityAction

Conclusion

- Discovering, documenting design regularities
  - scaleable to large industrial software systems
- Requires techniques that mine the source code for design regularities
- So that we can codify these regularities intensionally and co-evolve them with the source code
- Prior experience with aspect mining and first experiments make us hopeful that it can be done