Des analyses logicielles dédiées à base de modèles

Stéphane Ducasse - Usman Bhatti
RMoD - Synectique

http://www.synectique.eu
http://rmod.lille.inria.fr

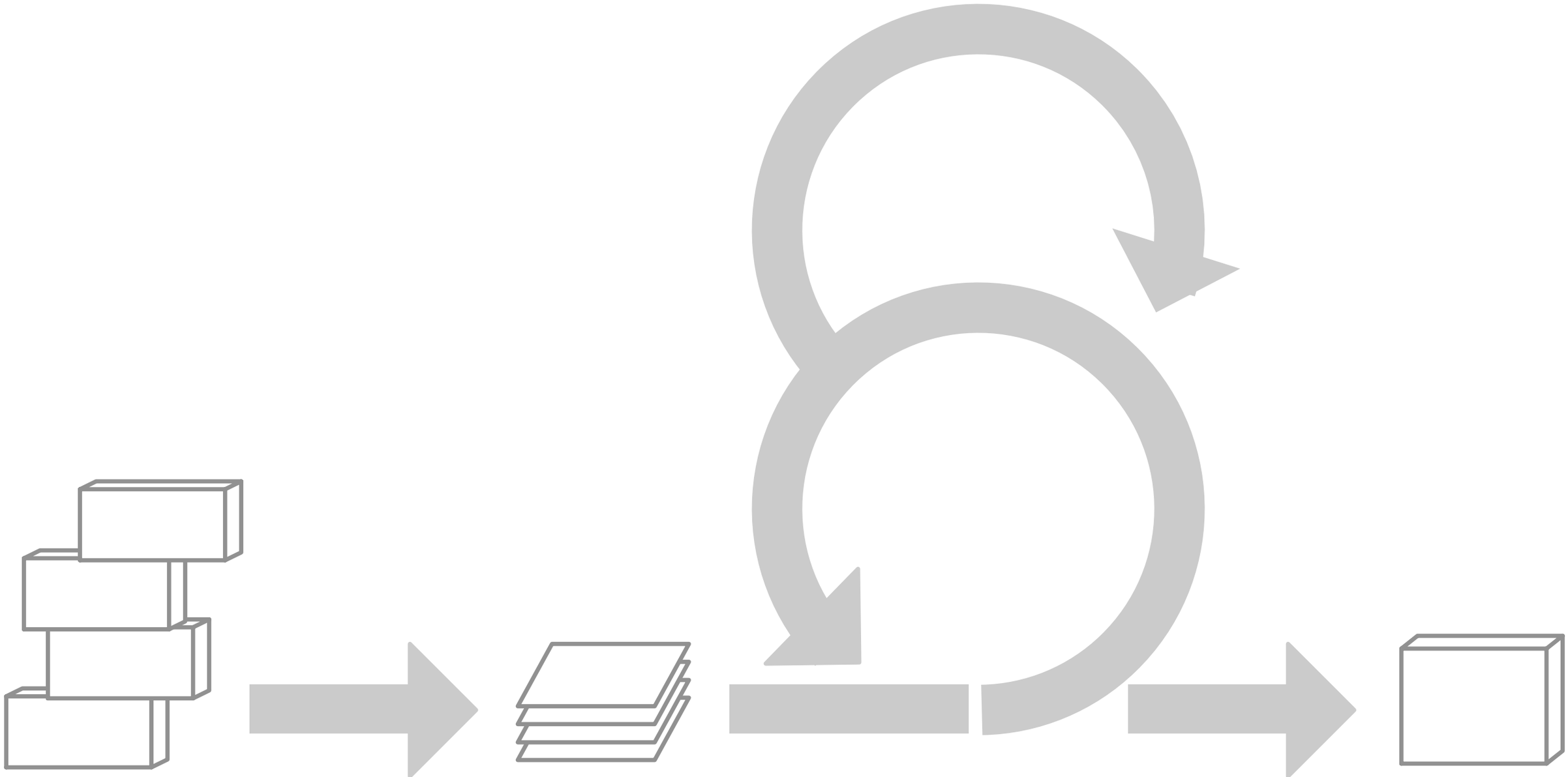
Un mot de présentation

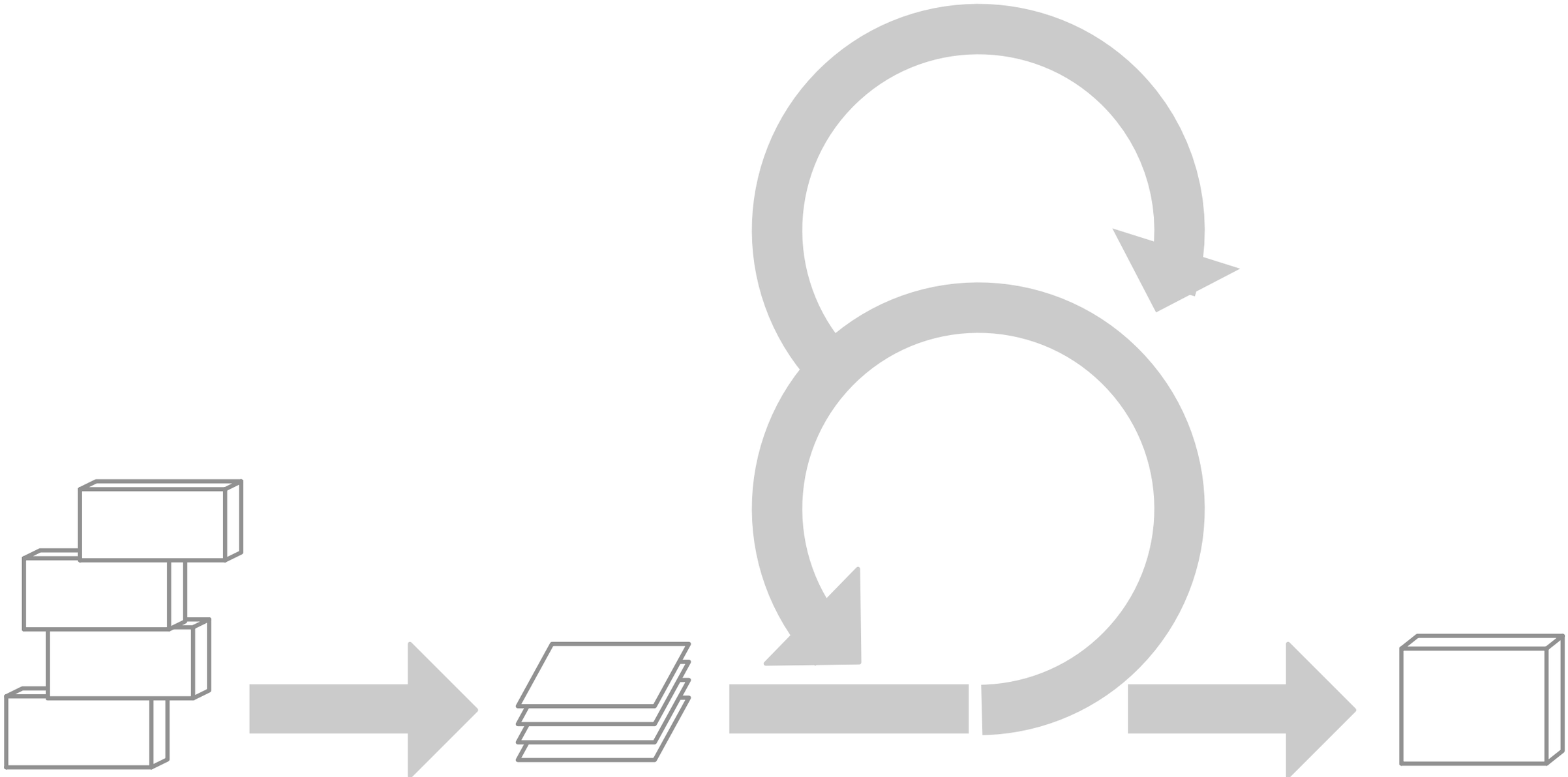
Depuis 1996 dans la maintenance et évolution logicielle

Auteur de Object-Oriented Reengineering Patterns

Une équipe avec plus 35 ans d'expérience cumulée en réingénierie

Contrôle processus industriels ?



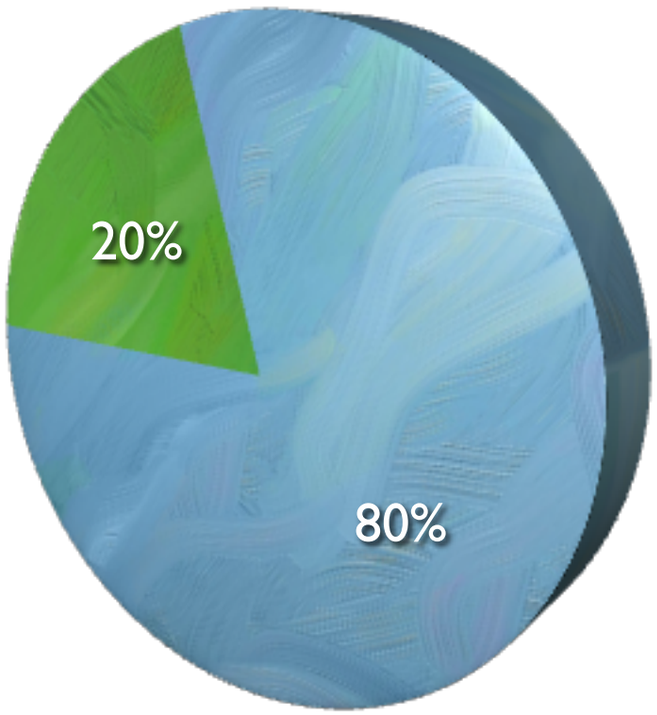


le retour est la clef

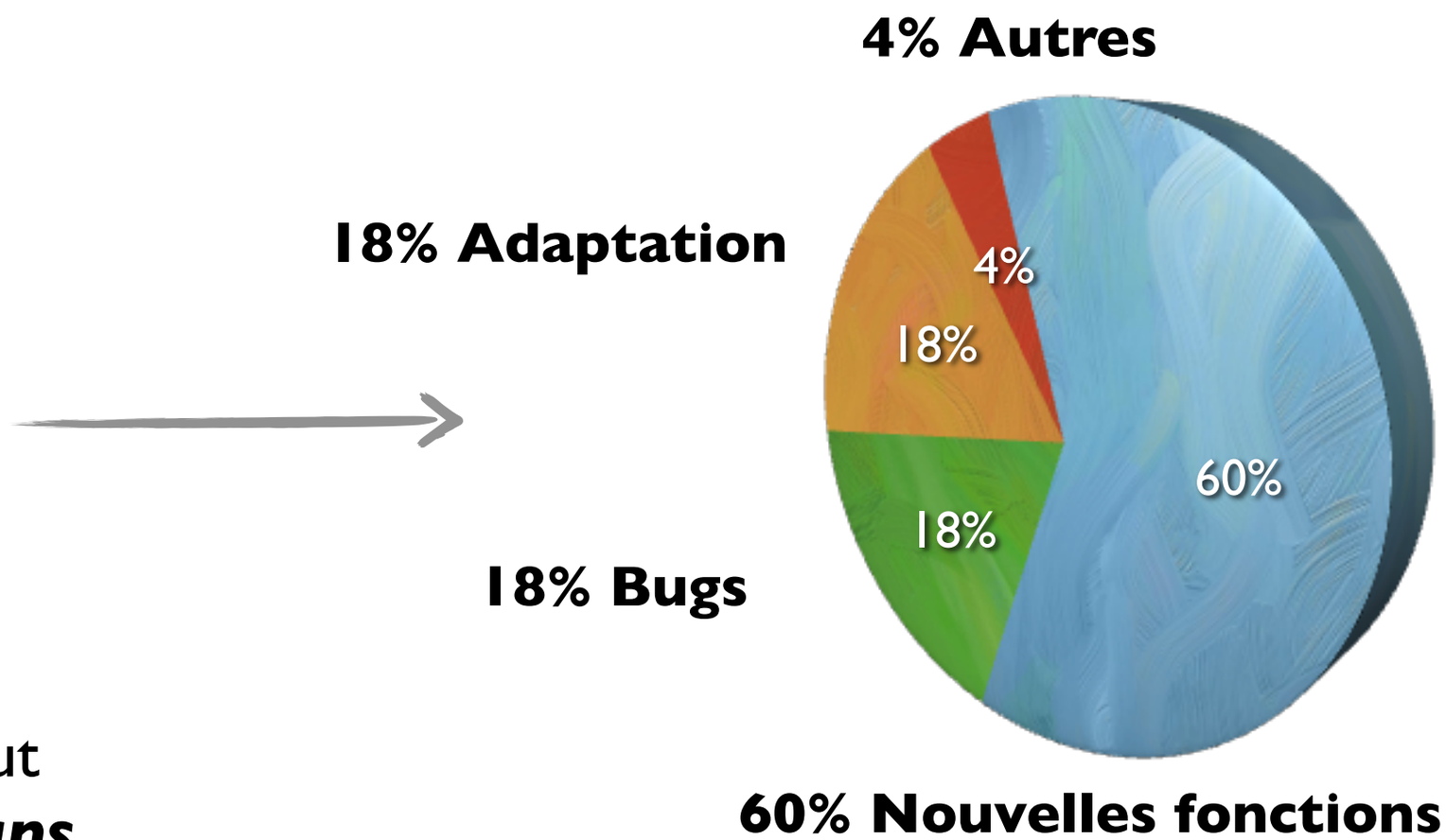
Mais du retour

- contextuel
- dédié
- continu

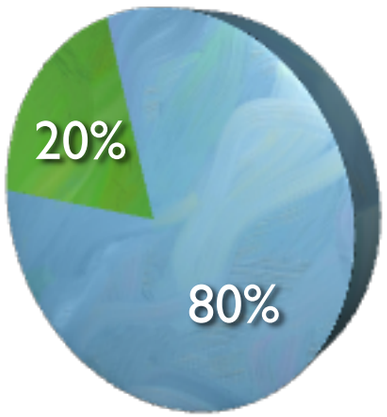
Analyse de Coût

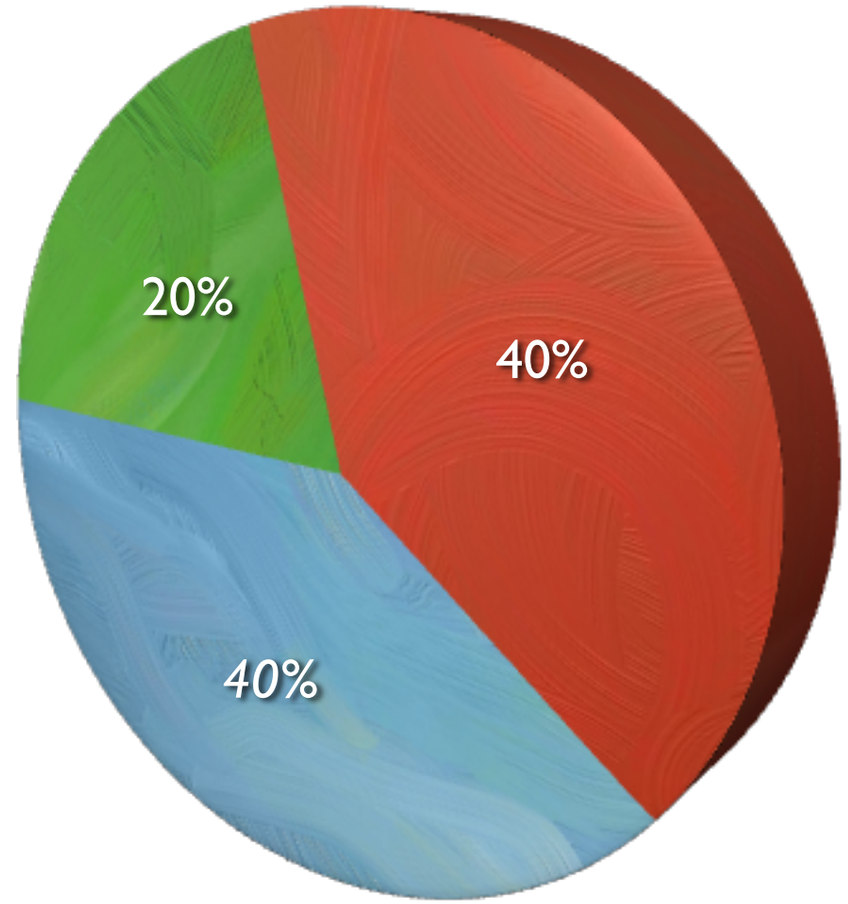


Entre **50**% and **80**% du cout
global total est passé dans l'évolution



50% du temps de developpement est passé à lire le code !

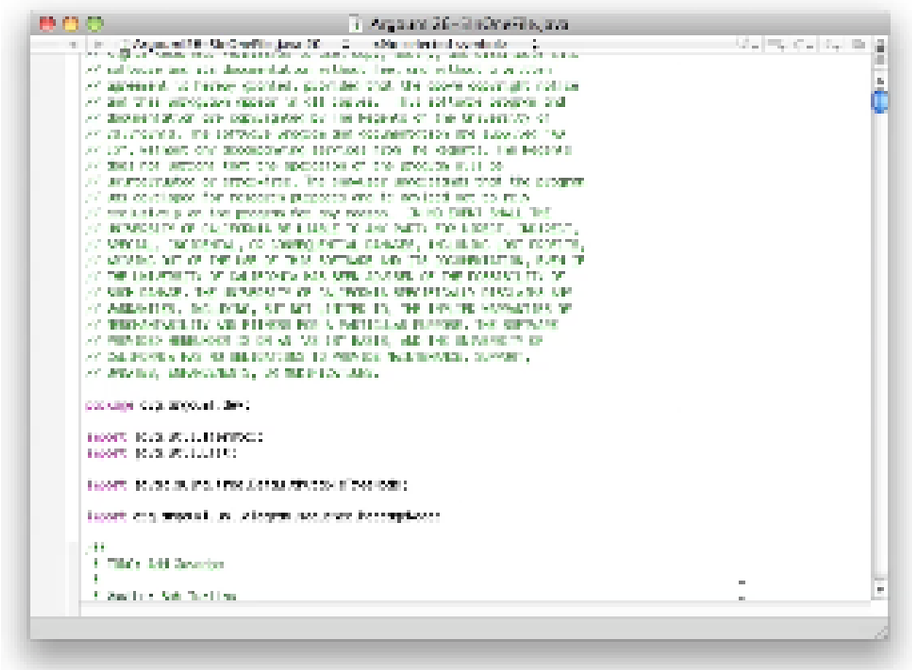




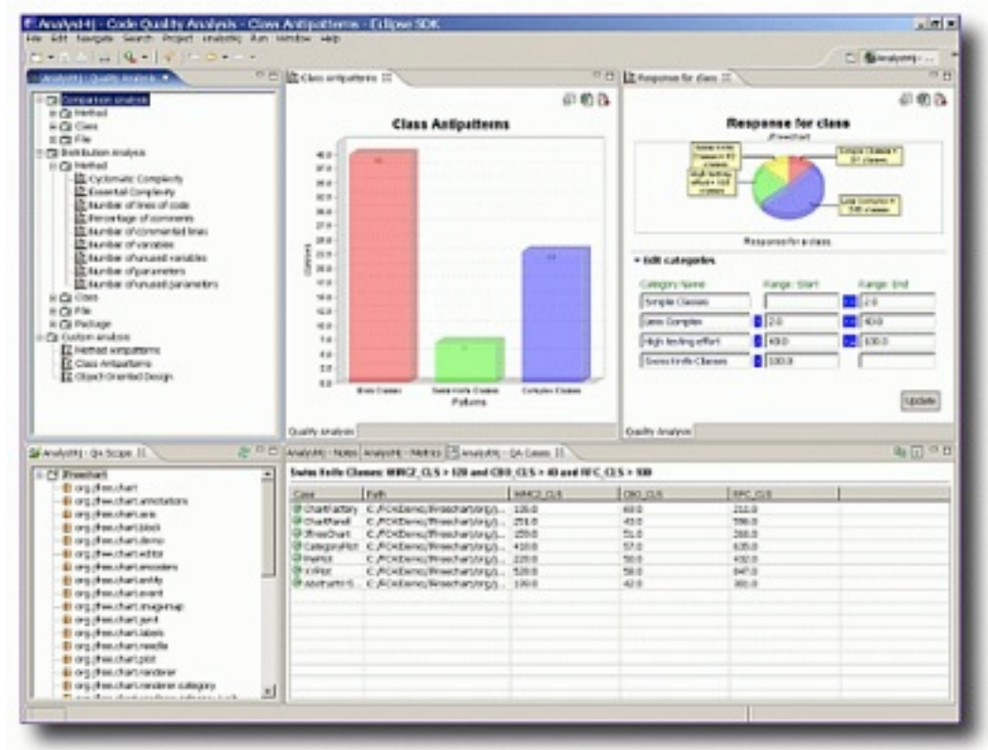
Entre **50**% and **80**% du cout ***global total est passé dans l'évolution***

énorme perte de temps avec des pratiques inadaptées et inefficaces

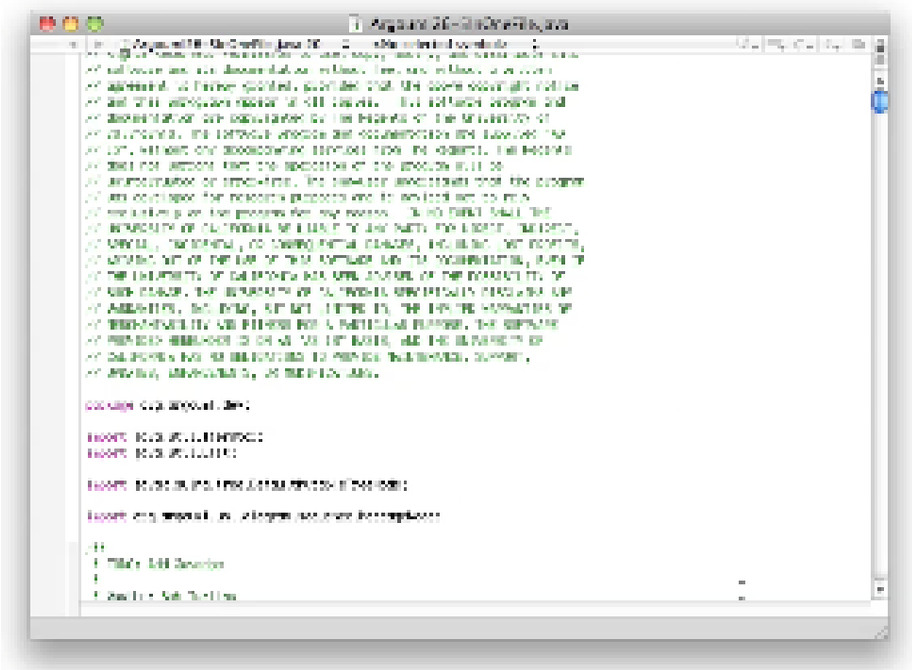
manuel
dédié



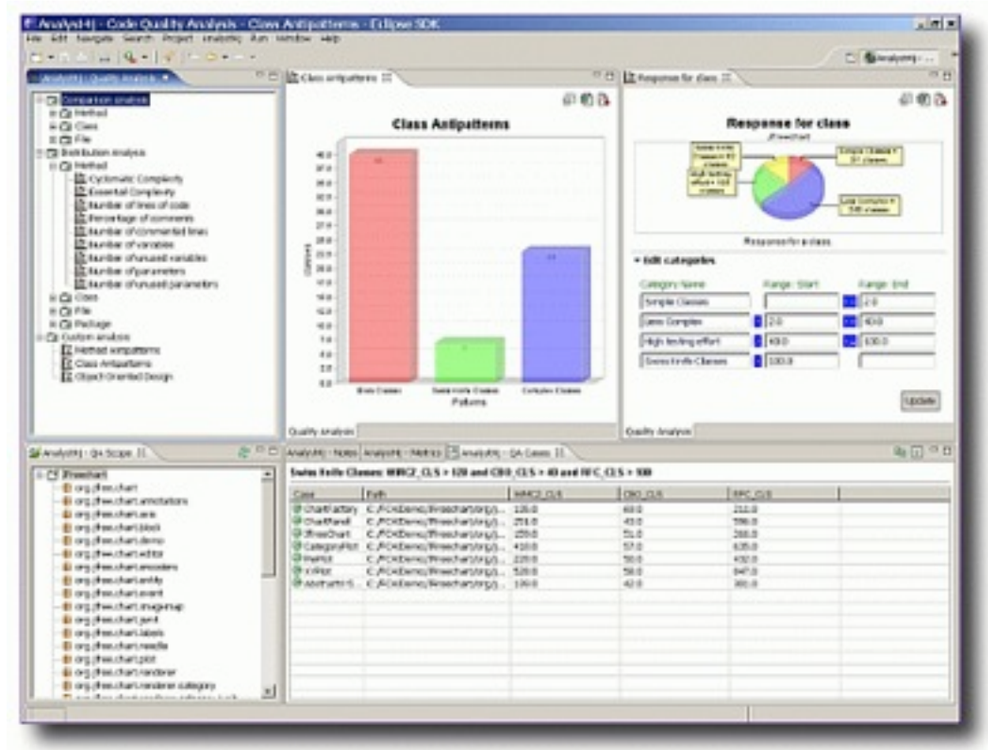
automatique
générique



~~manuel~~
dédié



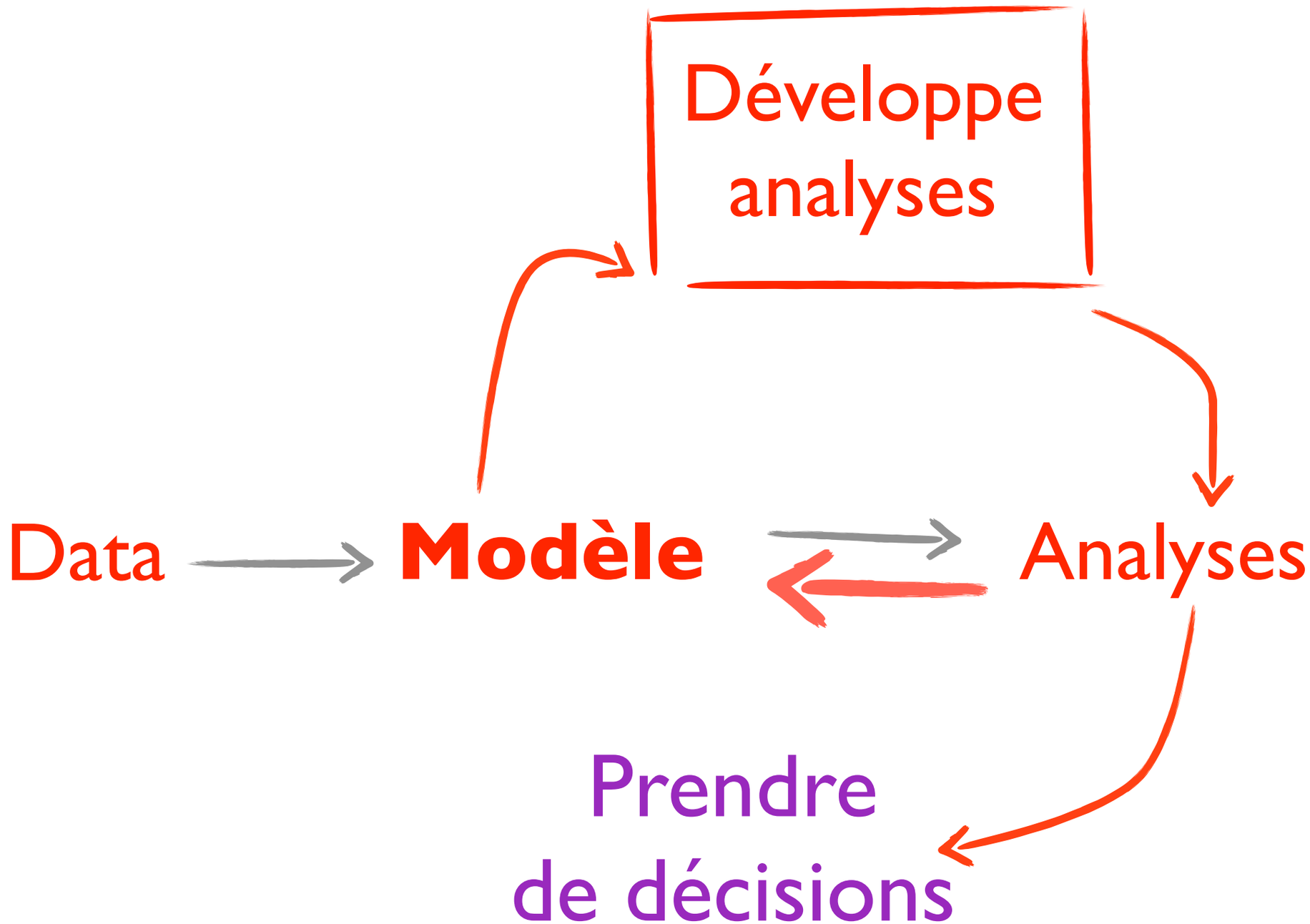
automatique
~~générique~~



Notre approche

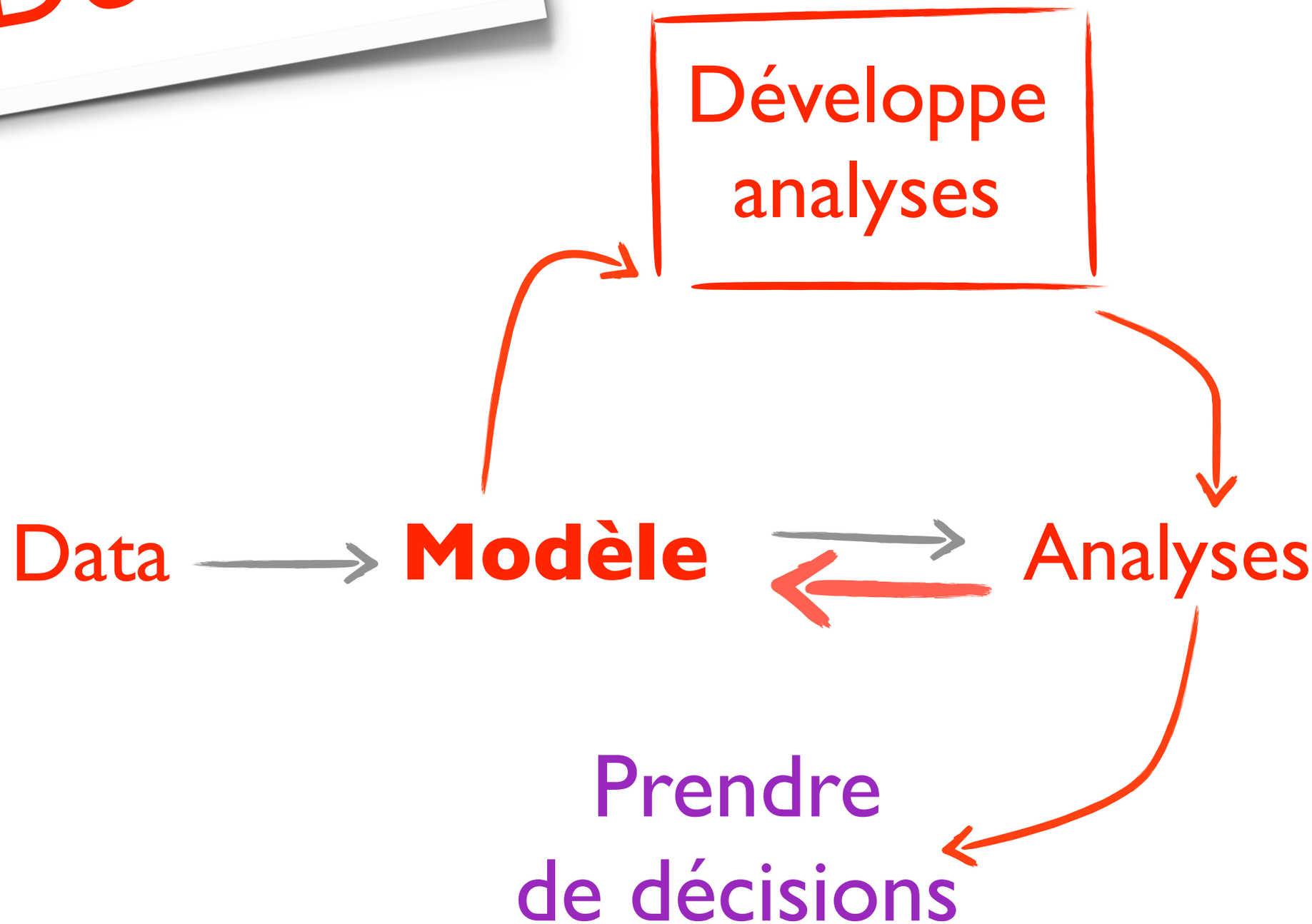
Des outils dédiés adaptés basés sur la définition ou enrichessement de modèles

Rentable en terme de coût



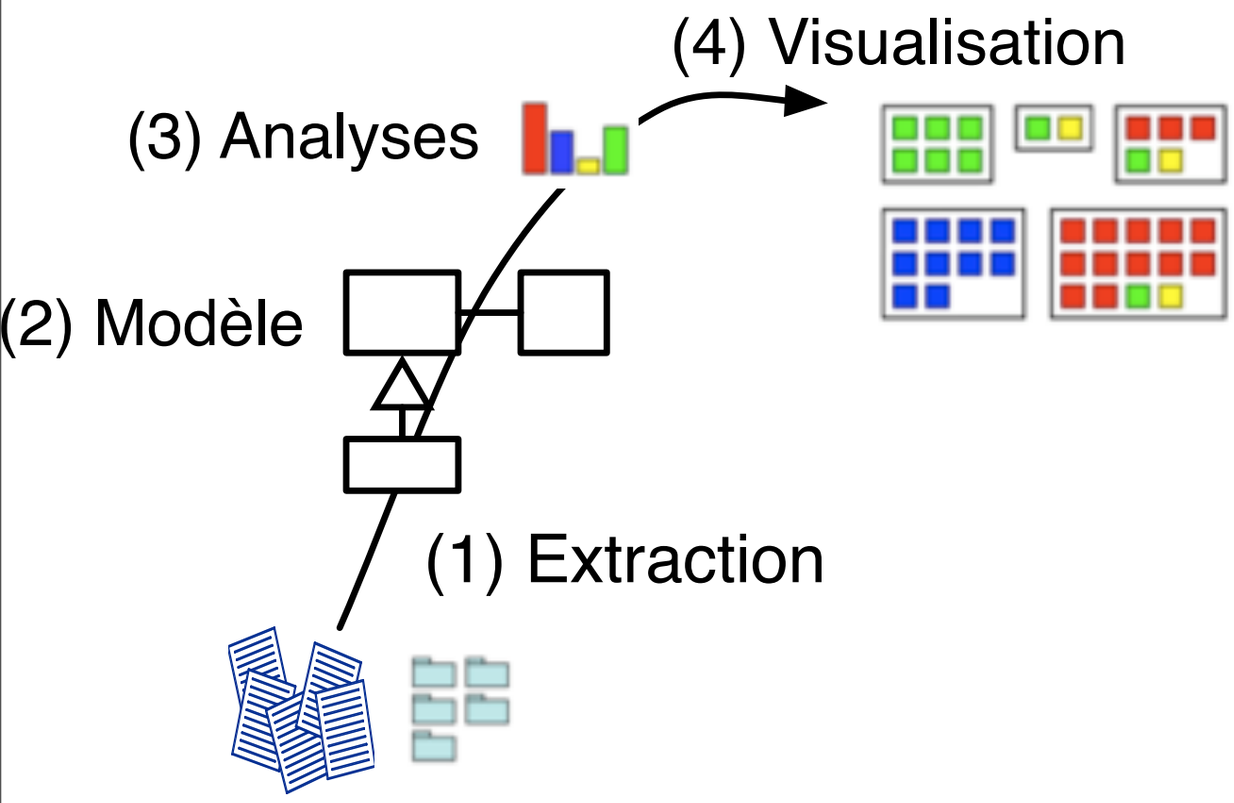
une analyse doit amener à une prise de décision

Outils Dédiés

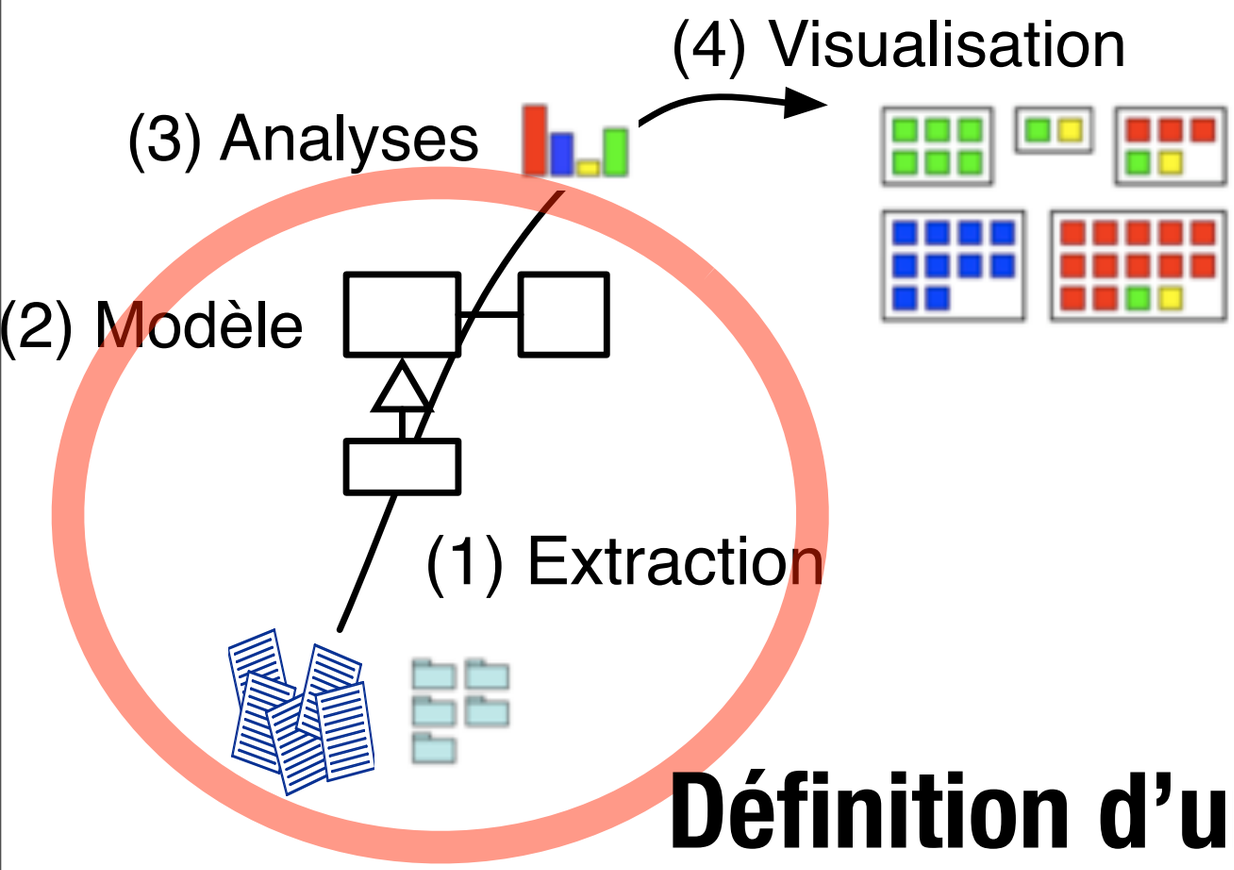


une analyse doit amener à une prise de décision

Exemple : qui est derrière le package X ?



Etape 1 - Création de modèle/Import

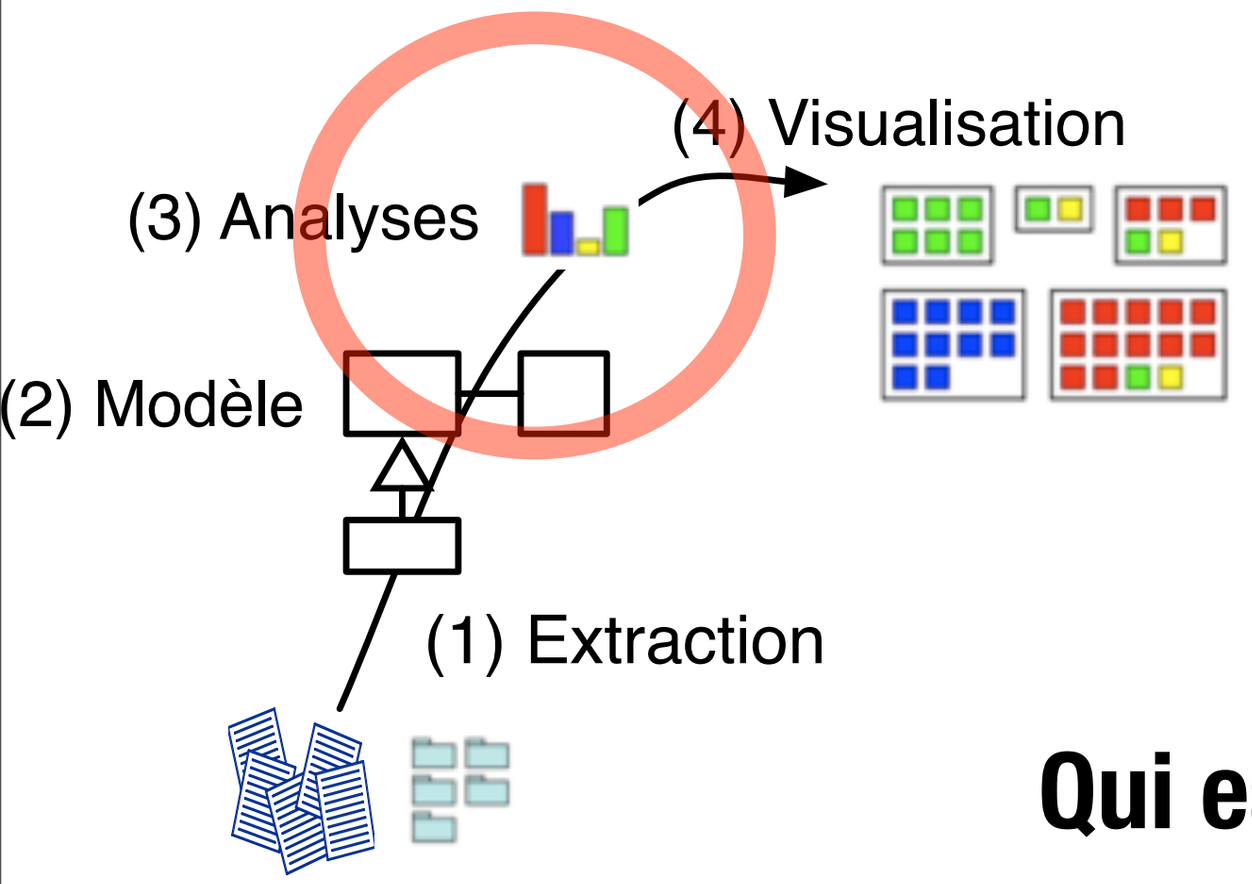


Définition d'un modèle d'auteurs, propriétés

Intégration au modèle de code

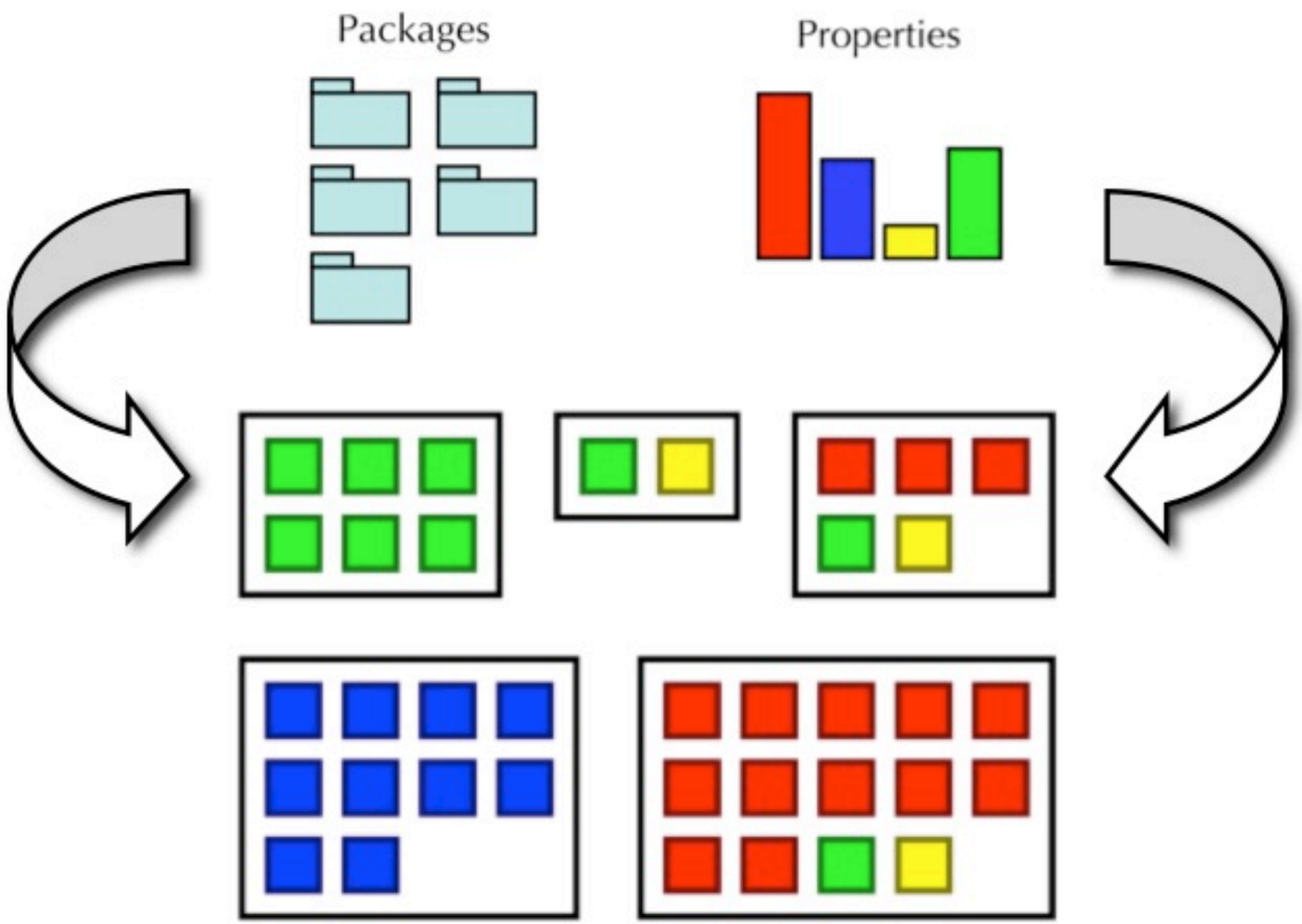
Extraction de données (CVS...)

Etape 2 - Analyses



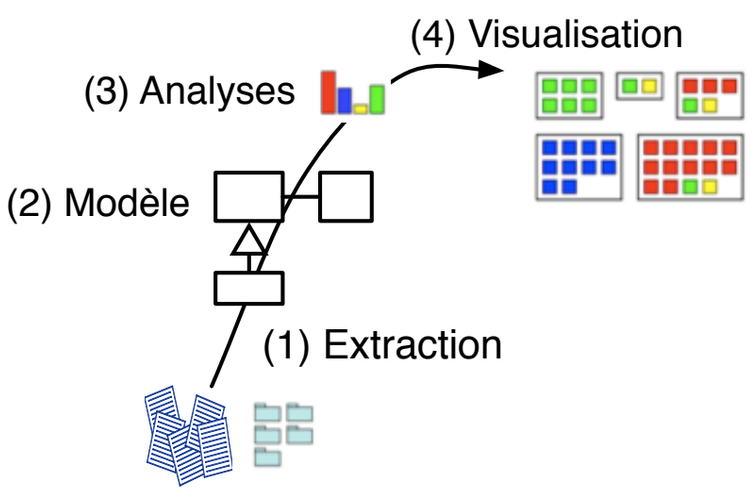
Qui est auteur de combien de lignes?

Etape : 3 - Création de Cartes et Outils

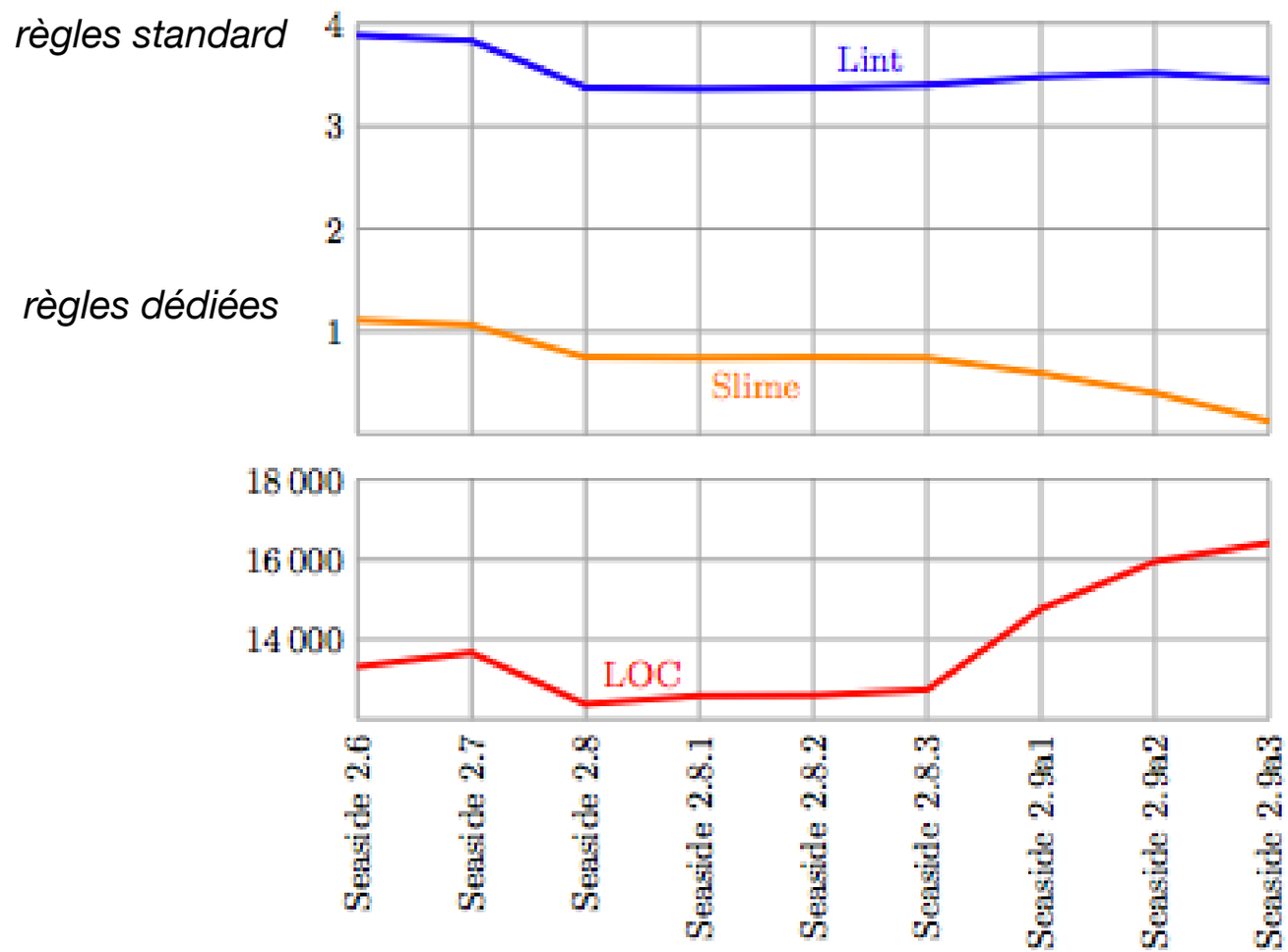


JBoss en un clin d'oeil

- Outil interactif



Il est avantageux de faire des analyses dédiées



Quid du coût ?

Vous payez déjà! (50% de la maintenance utilise des outils inéfficaces)

Problème : Depuis 15 ans la société X développe un environnement de formulaires pour de nombreux hopitaux.

Depuis **2 ans**, le logiciel souffre de problèmes de performances. Personne n'ose rien toucher!

Solution :

Création d'un modèle de trace

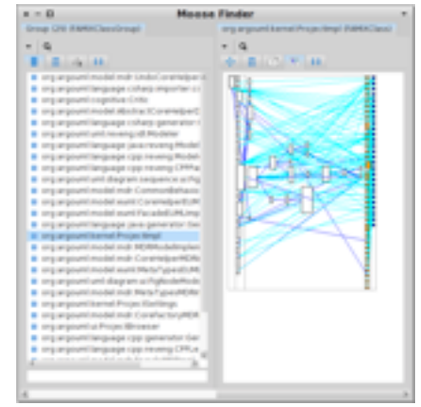
Construction d'un outil pour l'analyse des logs (**2 cycles de 6 semaines**)

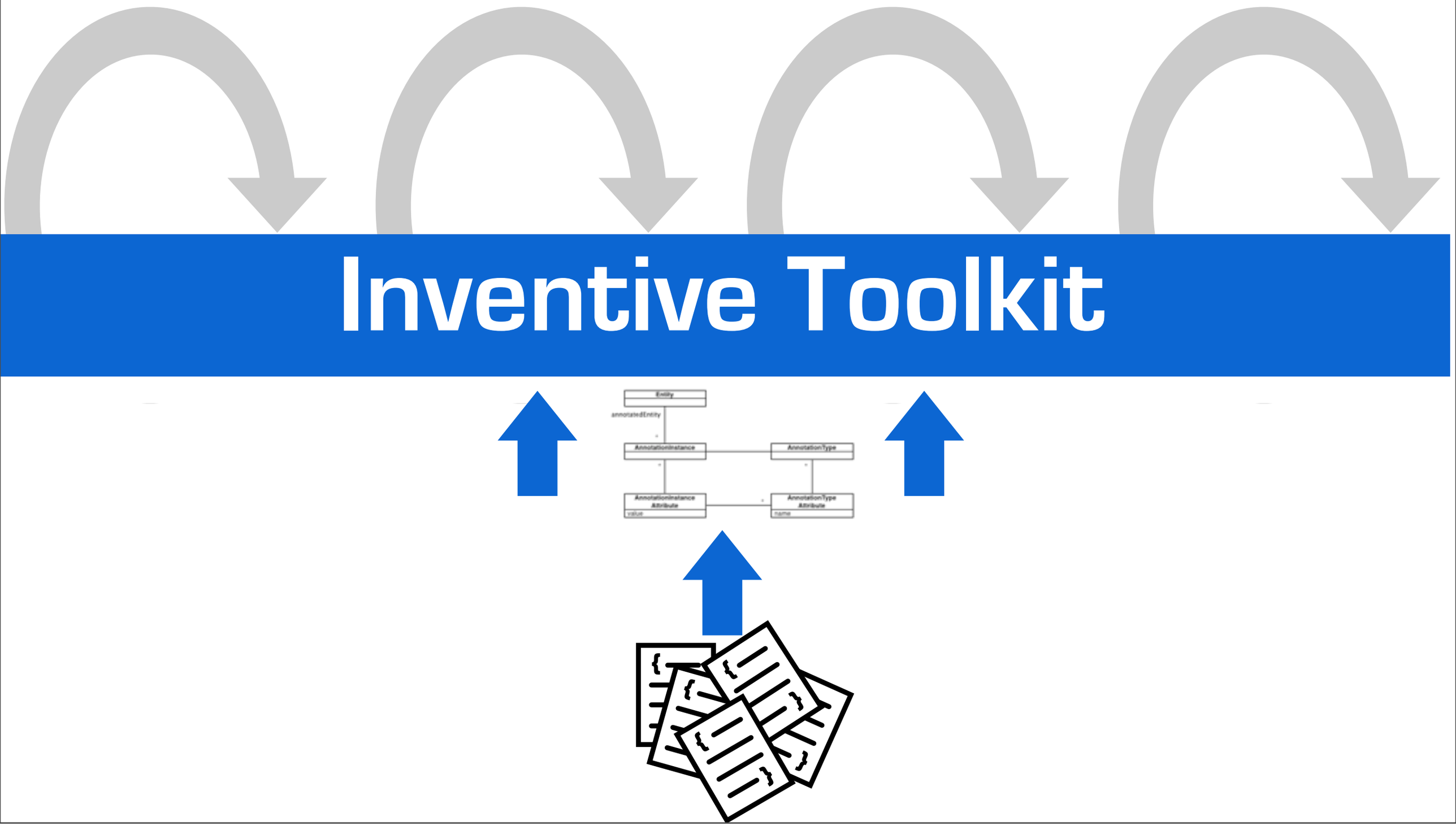
Former les ingénieurs et utiliser les résultats

classes select: #isGod

McCabe = 21

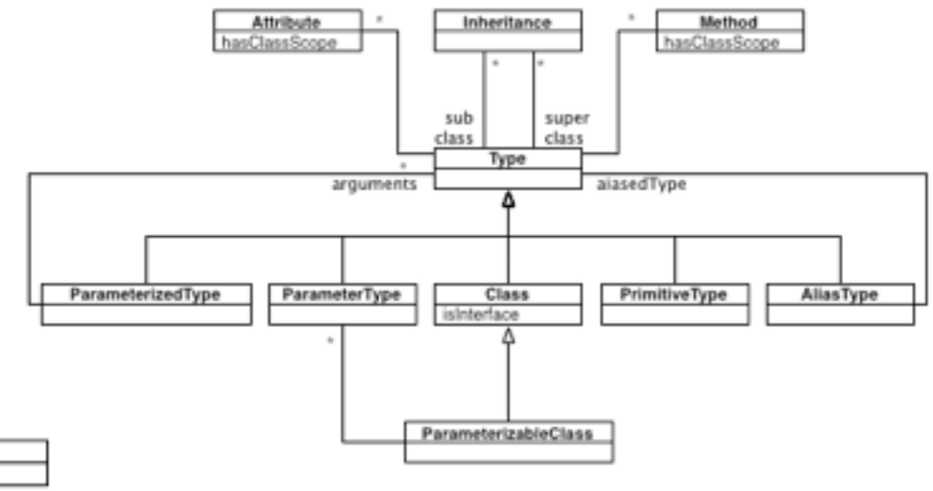
LOC = 753,000

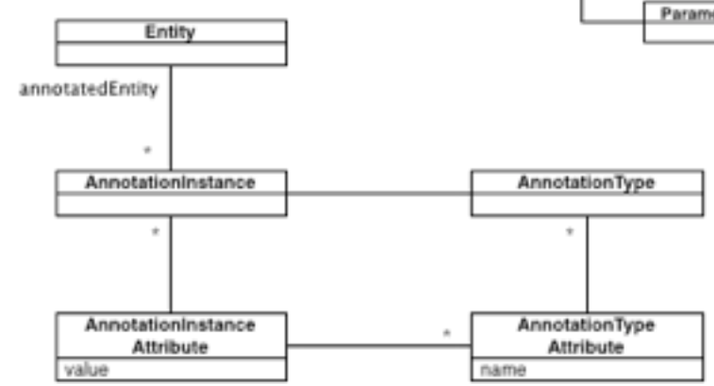


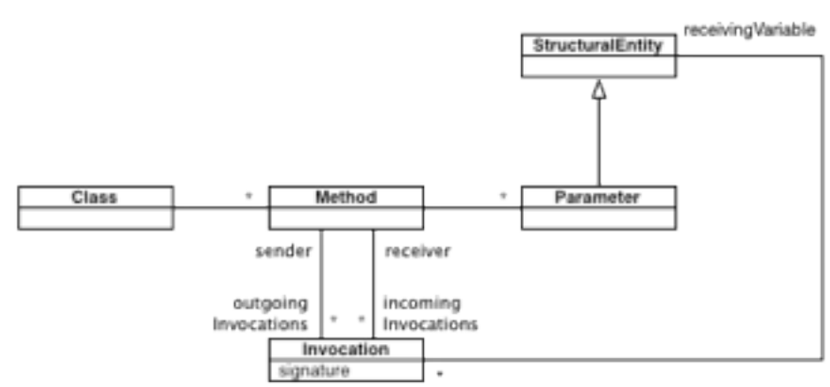


FAMIX

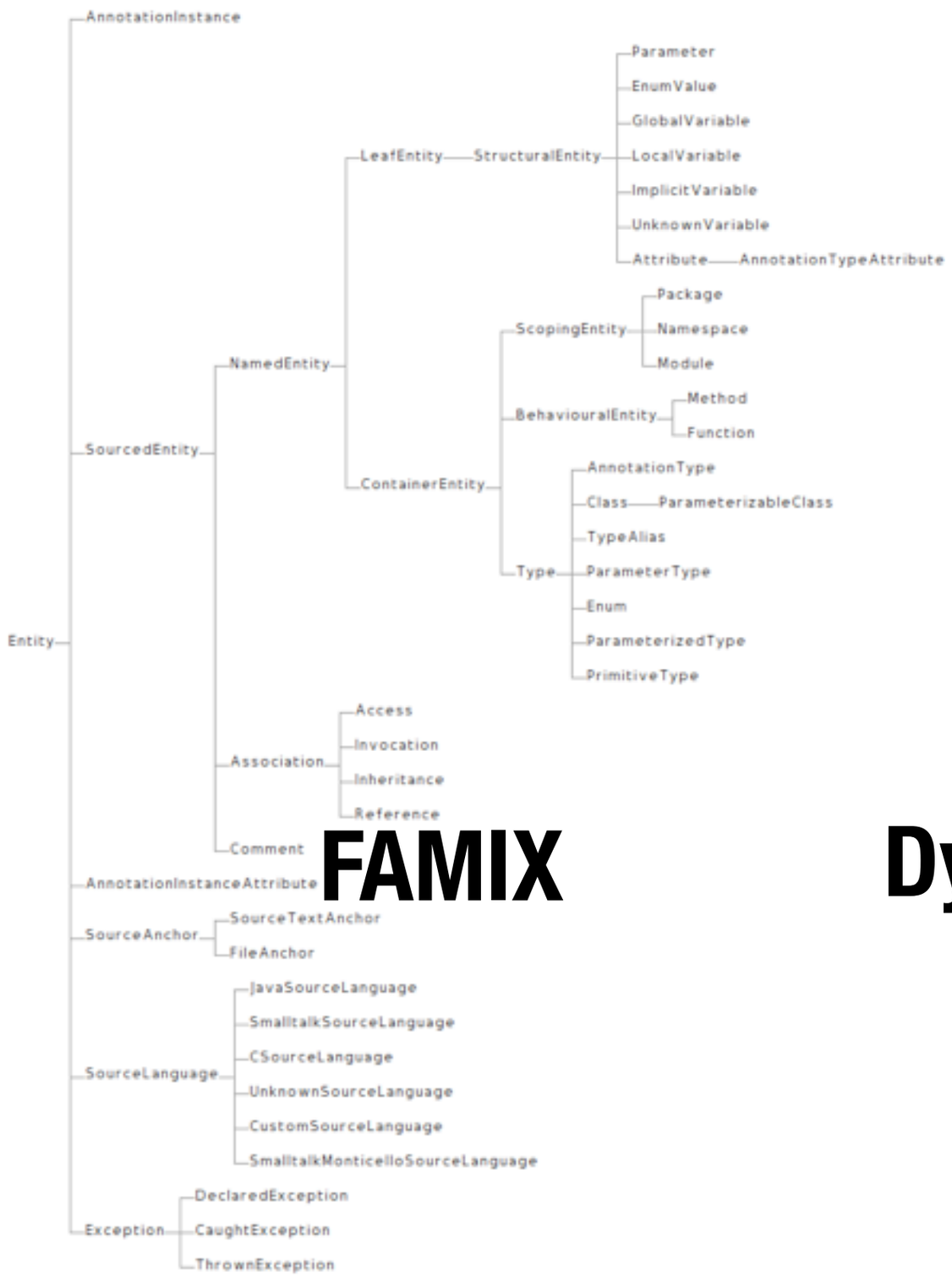
Entity
AnnotationInstance
SourcedEntity
NamedEntity
LeafEntity
StructuralEntity
Parameter
EnumValue
GlobalVariable
LocalVariable
ImplicitVariable
UnknownVariable
Attribute
AnnotationTypeAttribute
ContainerEntity
ScopingEntity
Package
Namespace
Module
BehaviouralEntity
Method
Function
Type
AnnotationType
Class
ParameterizableClass
TypeAlias
ParameterType
Enum
ParameterizedType
PrimitiveType
Association
Access
Invocation
Inheritance
Reference
Comment
AnnotationInstanceAttribute
SourceAnchor
SourceTextAnchor
FileAnchor
SourceLanguage
JavaSourceLanguage
SmalltalkSourceLanguage
CSourceLanguage
UnknownSourceLanguage
CustomSourceLanguage
SmalltalkMonticelloSourceLanguage
Exception
DeclaredException
CaughtException
ThrownException


Attribute
hasClassScope
Inheritance
Method
hasClassScope
sub class
super class
Type
arguments
aliasedType
ParameterizedType
ParameterType
Class
isInterface
PrimitiveType
AliasType
ParameterizableClass


Entity
annotatedEntity
AnnotationInstance
AnnotationType
AnnotationInstance Attribute
value
AnnotationType Attribute
name


StructuralEntity
receivingVariable
Class
Method
Parameter
sender
receiver
outgoing Invocations
incoming Invocations
Invocation
signature

FAMIX : a Family of Meta Models



FAMIX

Hismo (history)

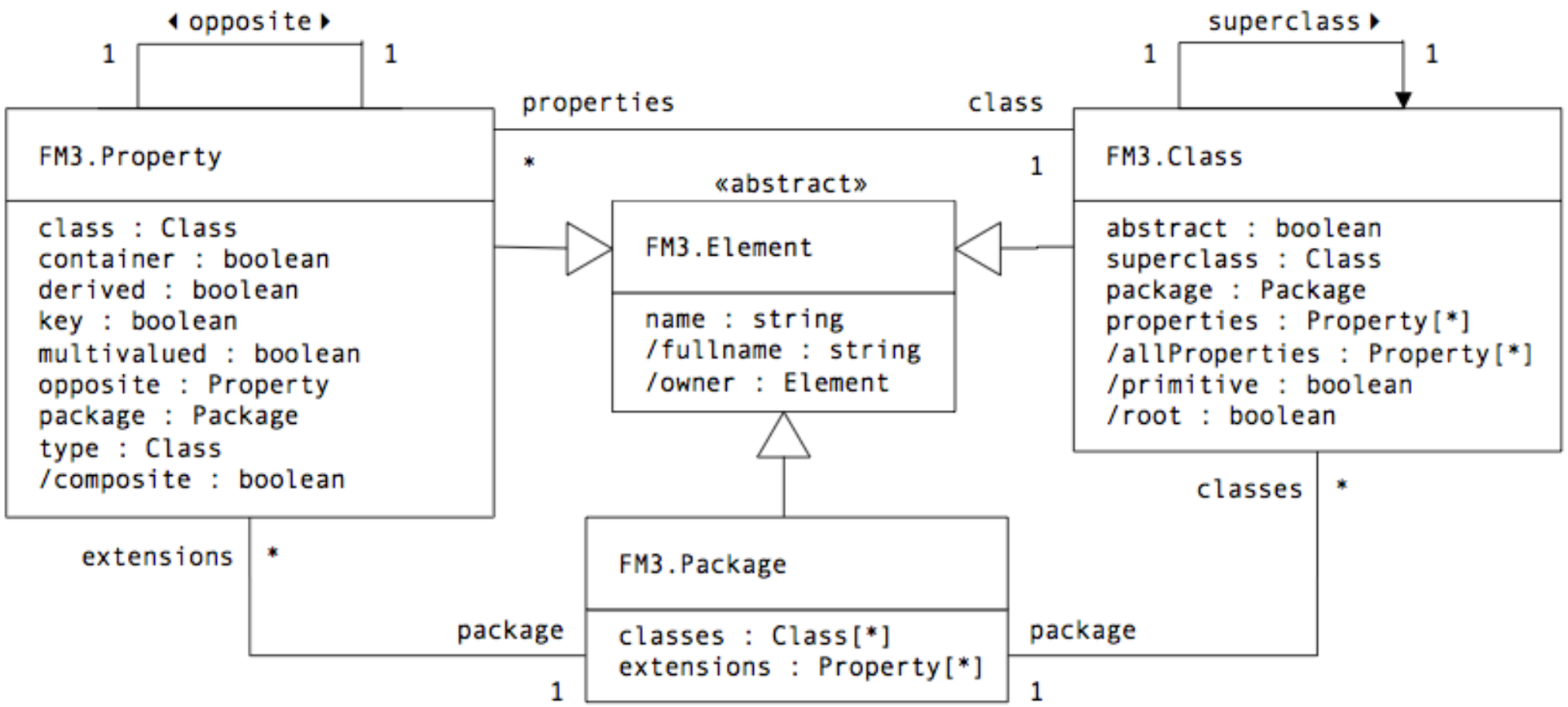
Dynamix

BugMaps

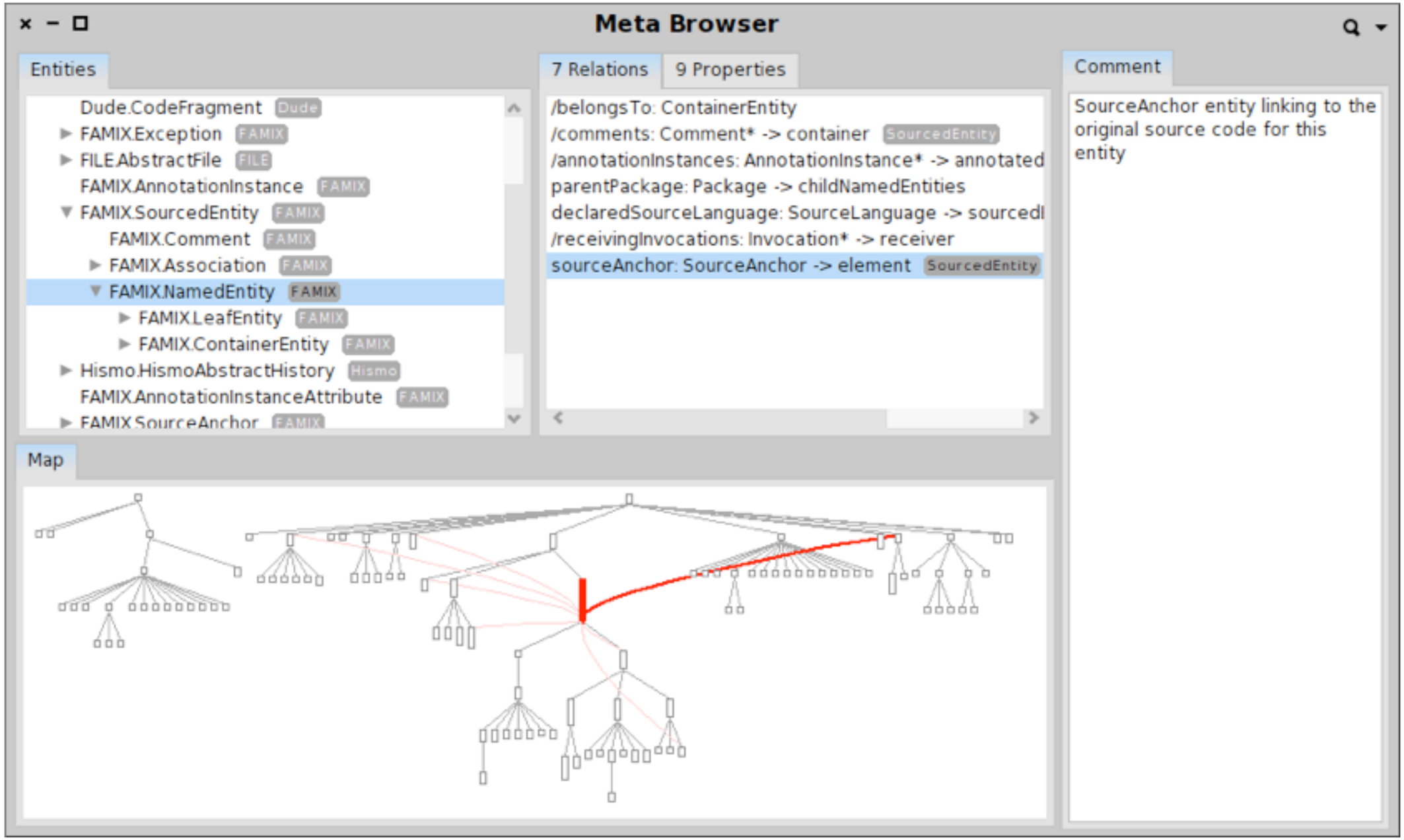
Authors

...

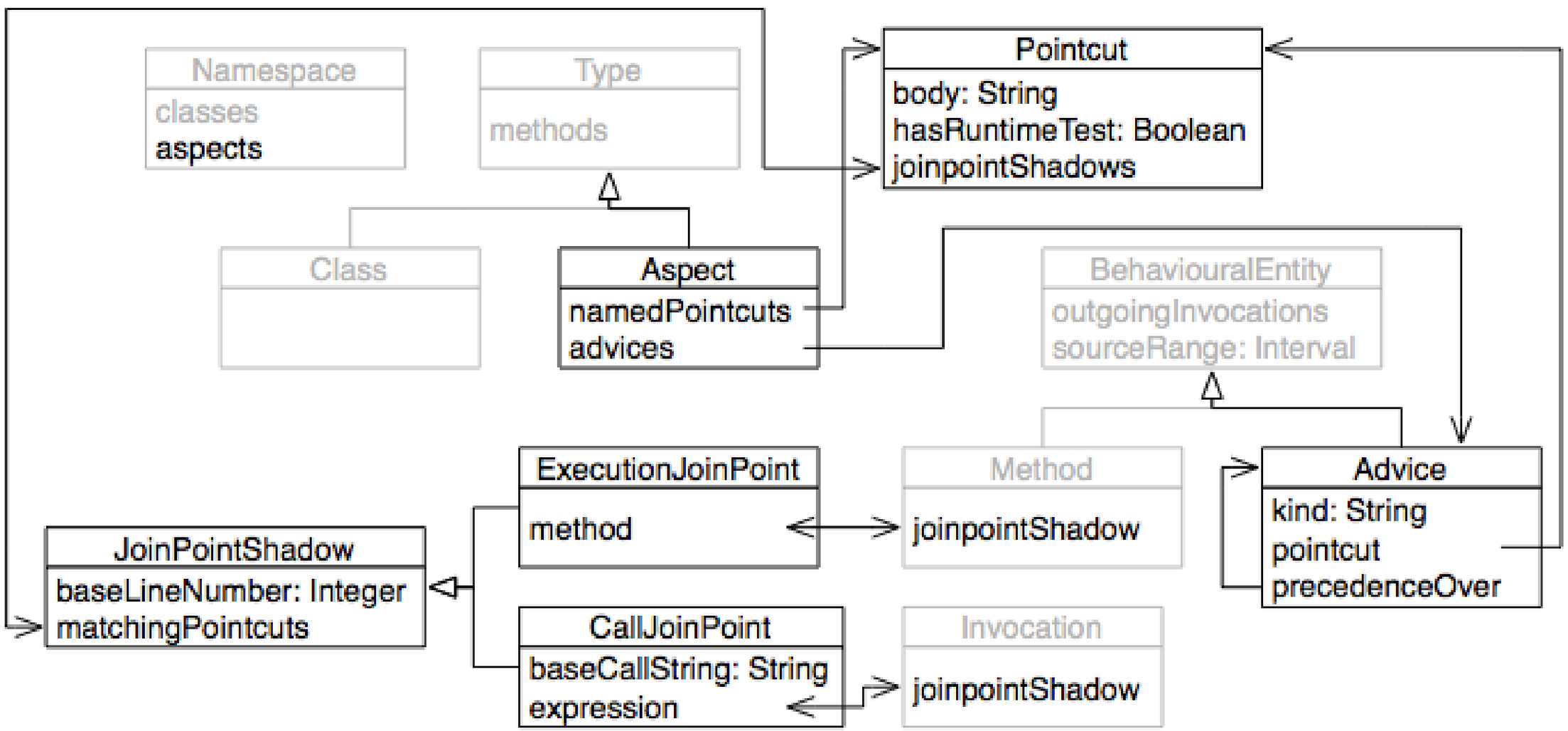
FAMIX Models are described using FAME



FAMIX Models are described using FAME



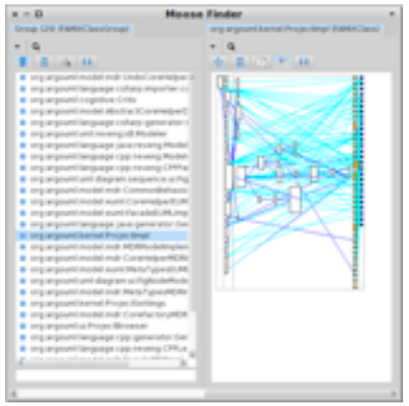
AspectMaps: Extending FAMIX

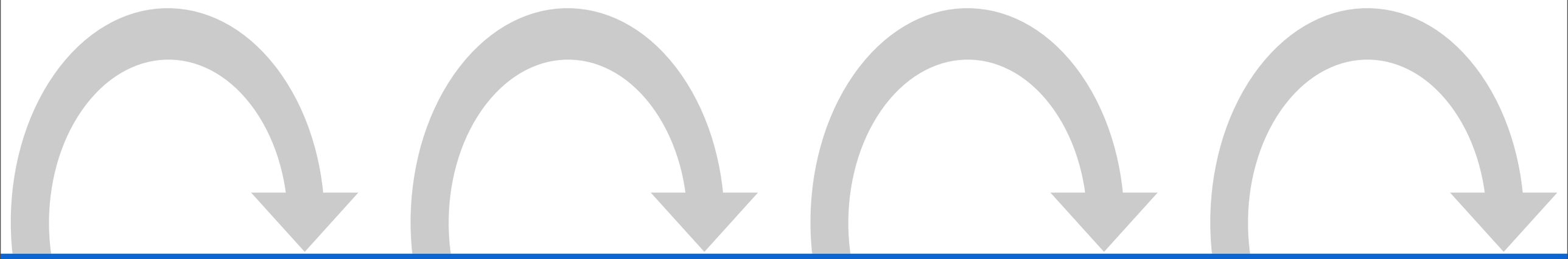


classes select: #isGod

McCabe = 21

LOC = 753,000





Inventive Toolkit

Métriques logiciels (best of)

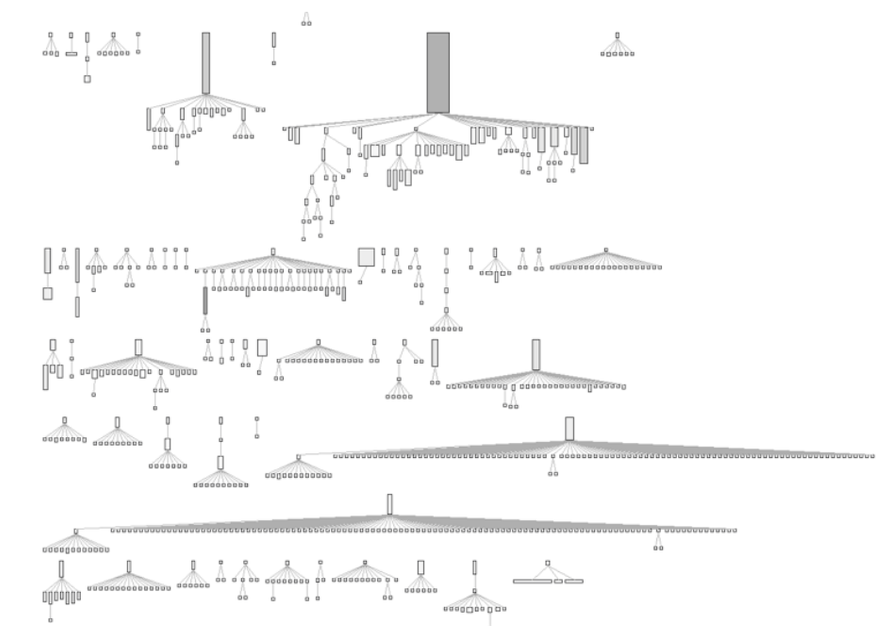
Modèles de qualité

ISO 9126, Squale (PSA-AirFrance)

Adaptation rapide

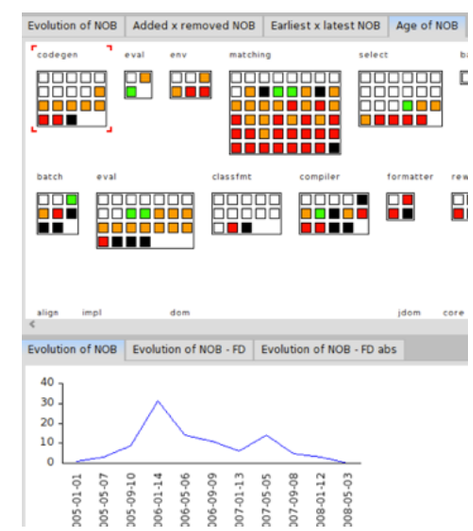
Définition spécifique au business

Cartes *dédiées*
pour *la software business intelligenc*



System Complexity

Carte de Distribution



...

Queries dans un contexte

Moose Panel

ASTCore

All model classes (70) (FAMIXClassGroup)

Smalltalk::RBRootScope
Smalltalk::RBPatternWrapperBlockNode
Smalltalk::RBPatternScanner
Smalltalk::RBFormatter
Smalltalk::RBAssignmentNode
Smalltalk::RBTemporaryBinding
Smalltalk::RBSpecialBinding
Smalltalk::RBCascadeNode
Smalltalk::RBLiteralArrayToken
Smalltalk::RBPatternParser
Smalltalk::RBReadBeforeWrittenTester
Smalltalk::RBVariableBinding
Smalltalk::RBSmallDictionaryTest
Smalltalk::RBNodedScope
Smalltalk::RBInstanceBinding
Smalltalk::RBToken
Smalltalk::RBPatternMethodNode
Smalltalk::RBSelfBinding
Smalltalk::RBVariableNode
Smalltalk::RBPatternVariableNode
Smalltalk::RBFormatterTests
Smalltalk::RBUndeclaredVariableNotification

50 50 / 70

each numberOfLinesOfCode > 100

Group (16) (FAMIXClassGroup)

Smalltalk::RBSequenceNode
Smalltalk::RBScanner
Smalltalk::RBArrayNode
Smalltalk::RBBlockNode
Smalltalk::RBConfigurableFormatter
Smalltalk::RBSemanticTest
Smalltalk::RBMessageNode
Smalltalk::RBParser
Smalltalk::RBMethodNode
Smalltalk::RBFormatter
Smalltalk::RBAssignmentNode
Smalltalk::RBCascadeNode
Smalltalk::RBReadBeforeWrittenTester
Smalltalk::RBProgramNodeTest
Smalltalk::RBPragmaNode
Smalltalk::RBProgramNode

System Complexity Locator

System Complexity with 16 selected classes

RBProgramNodeVisitor RBScanner RBParser RBVariableBinding RBToken RBLexicalScope RBProgramNode

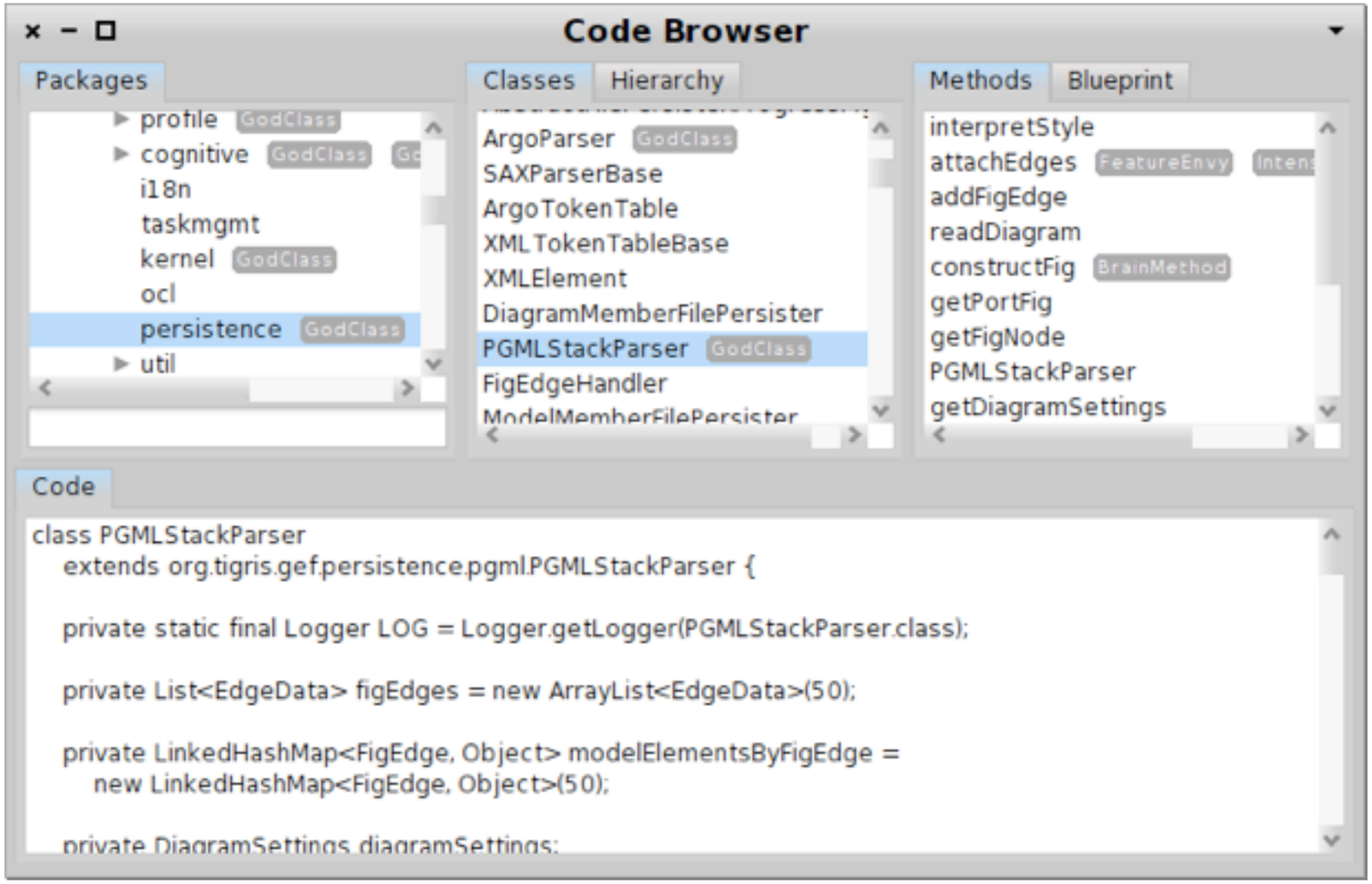
Group subtracted with Group (7 FAMIXClasses)

Outils spécifiques

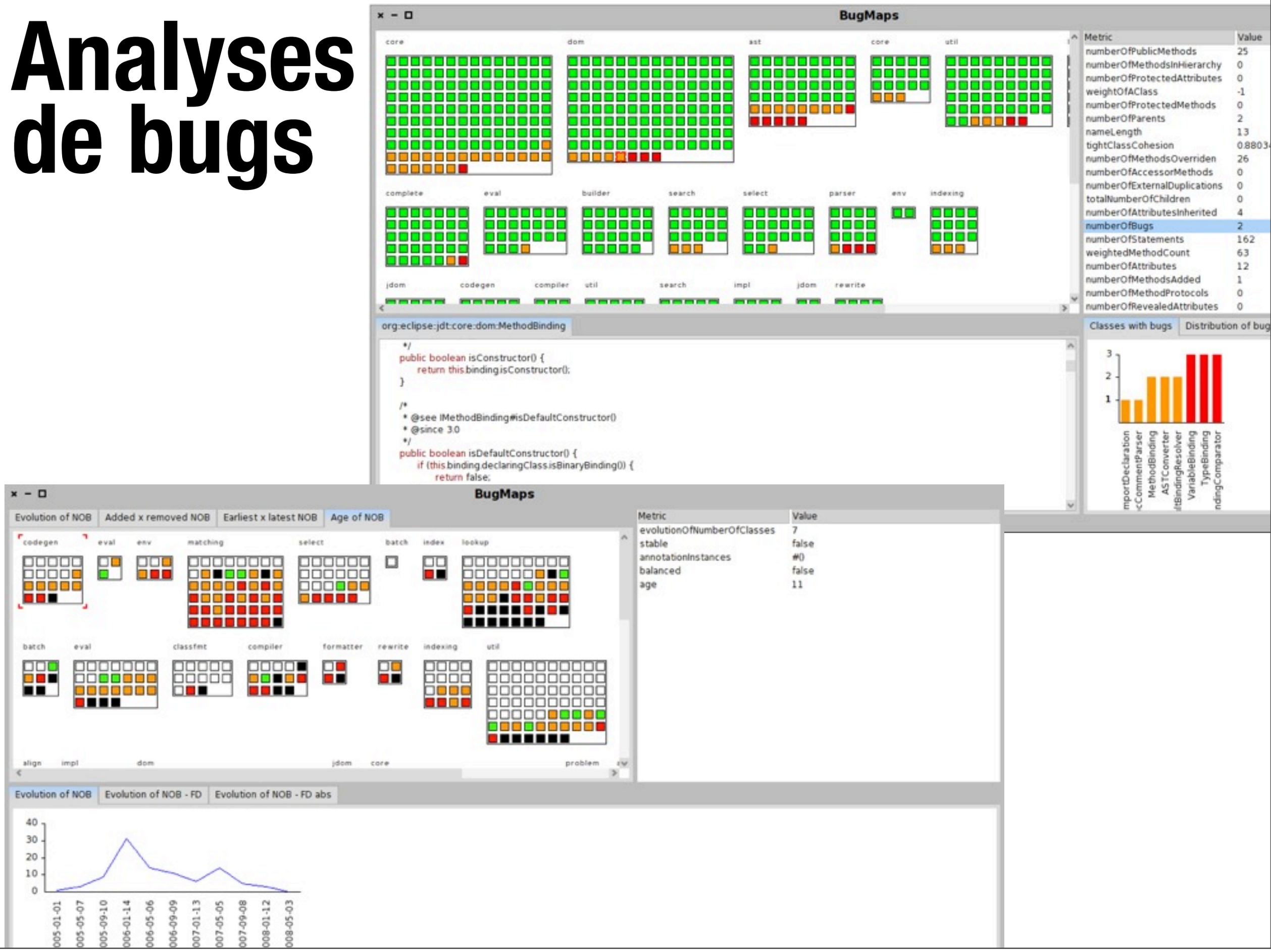
riches

compacts

meilleur focus

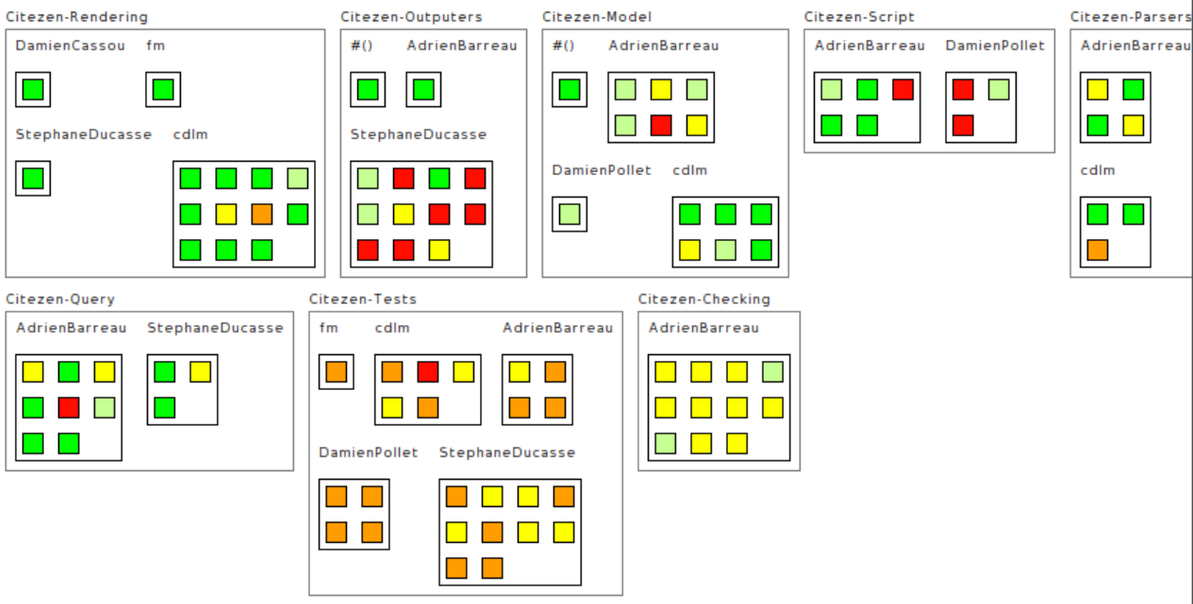


Analyses de bugs



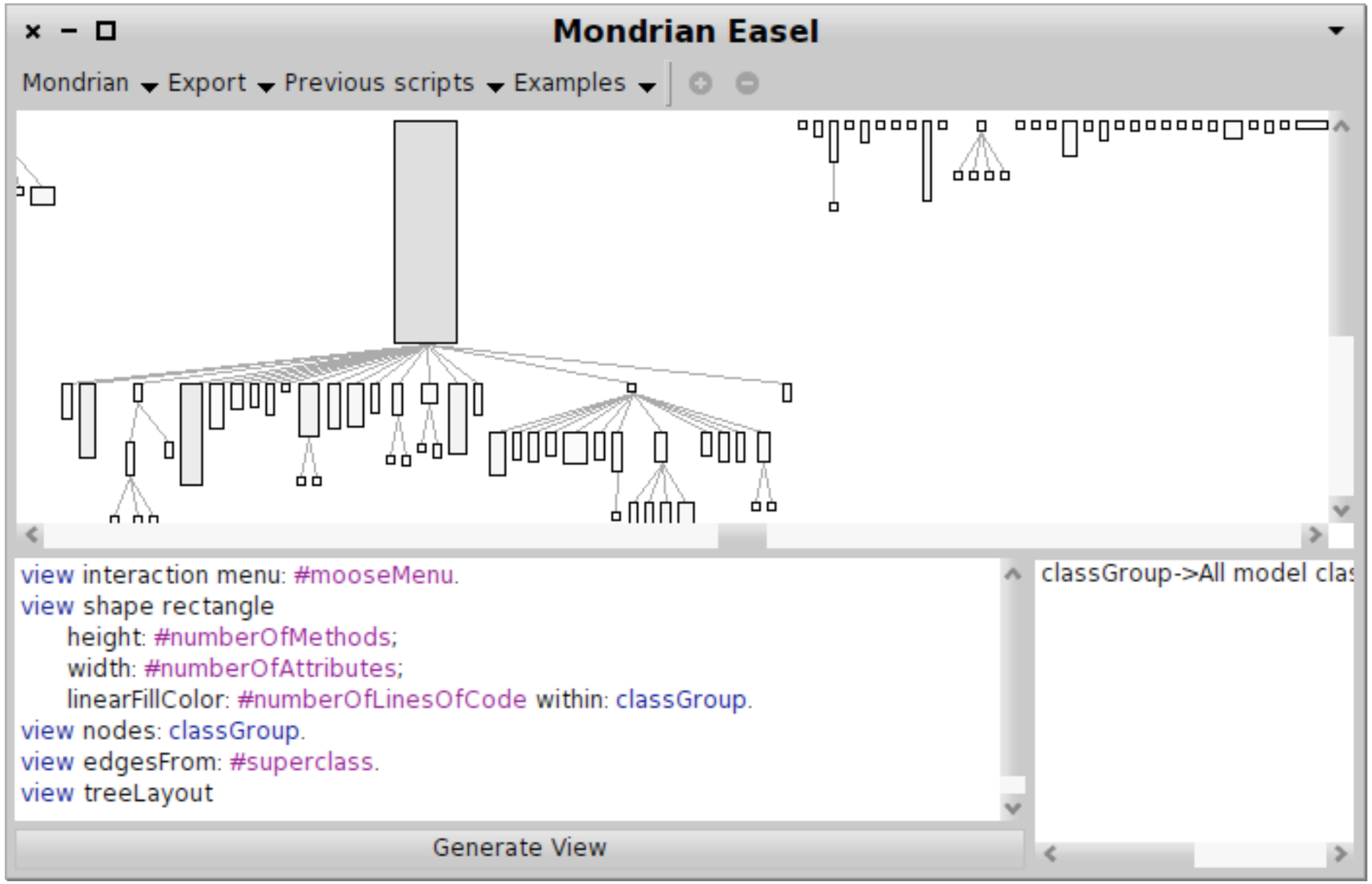
Aggrégation de données / ponts entre outils

parseur combinateur modulaire

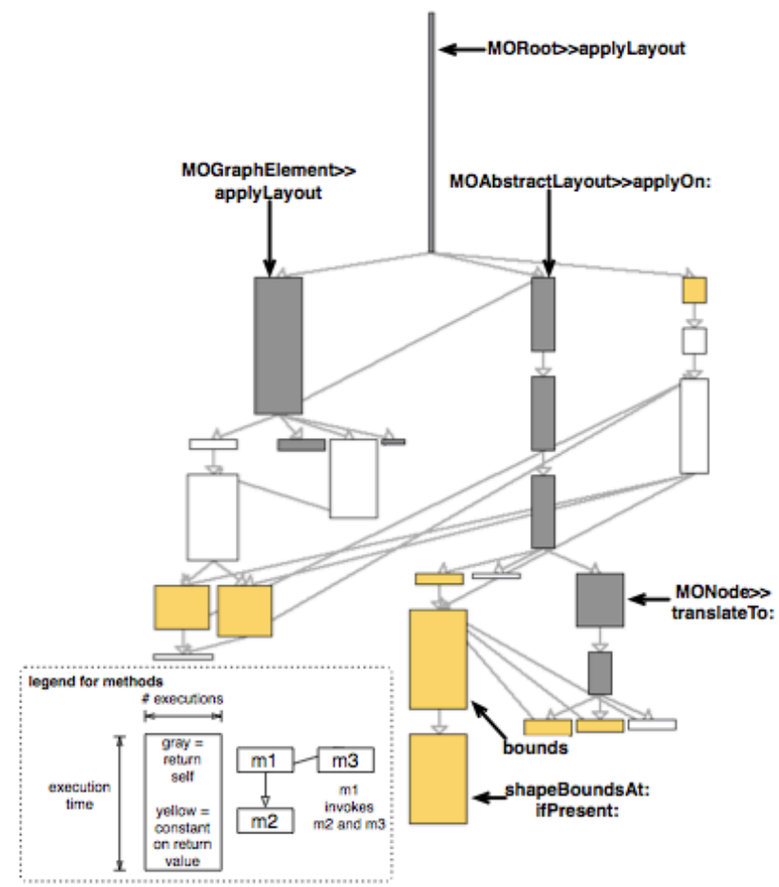


Exemple :
corréler les bugs et la couverture de tests

Analyses dédiées

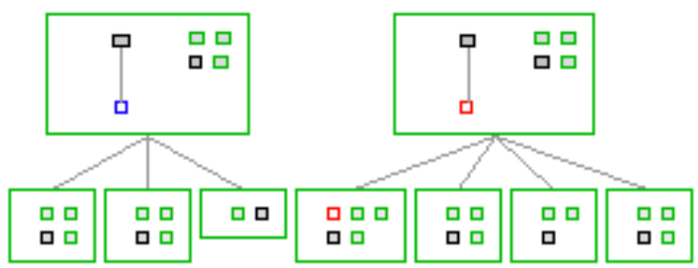


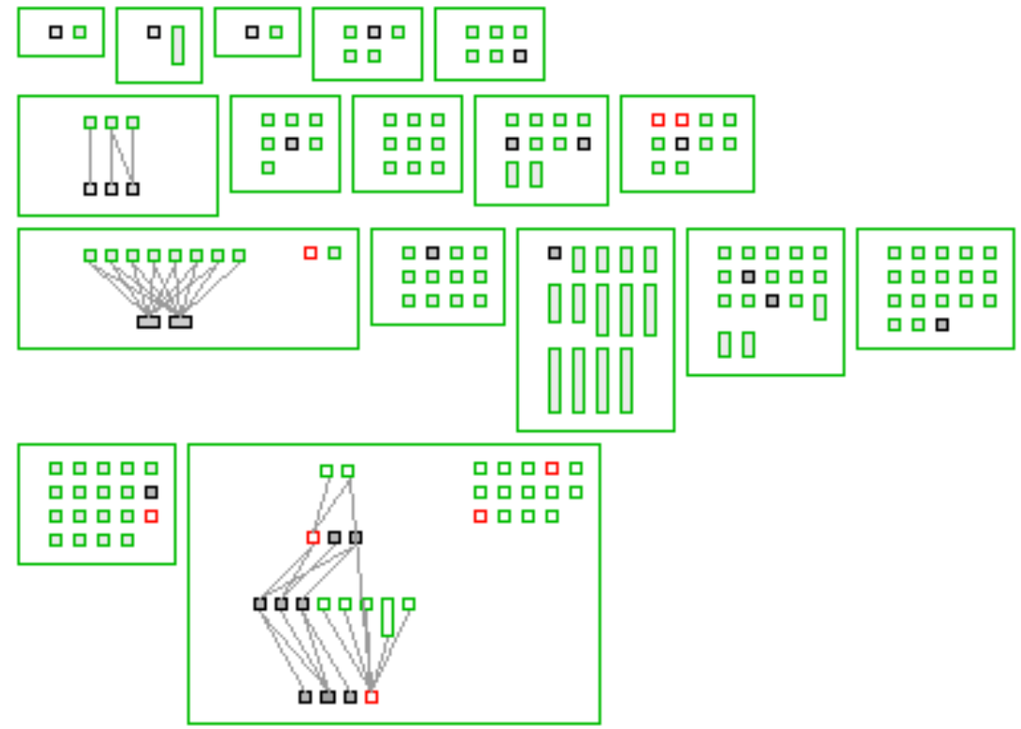
Logs et Performance

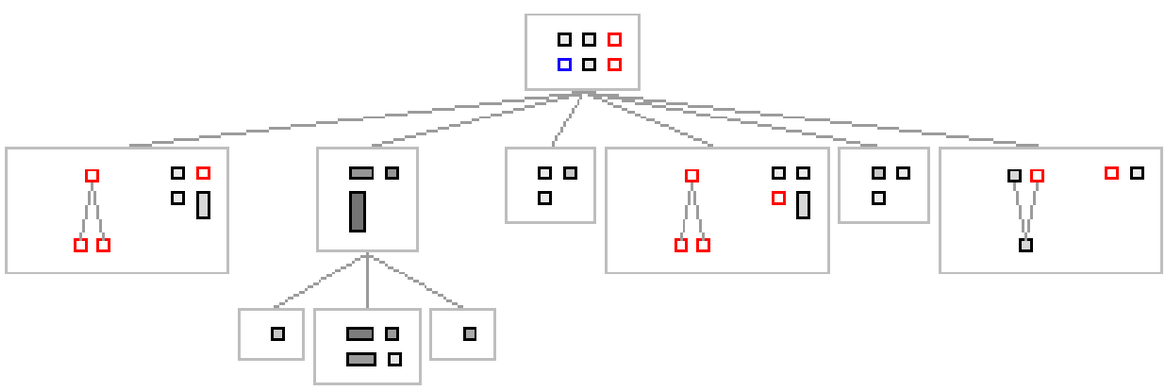


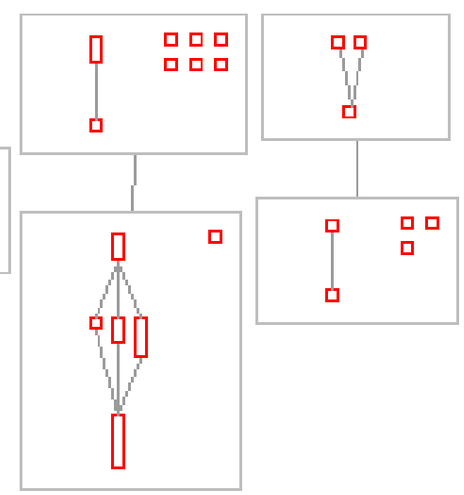
Mise en perspective de couverture de tests

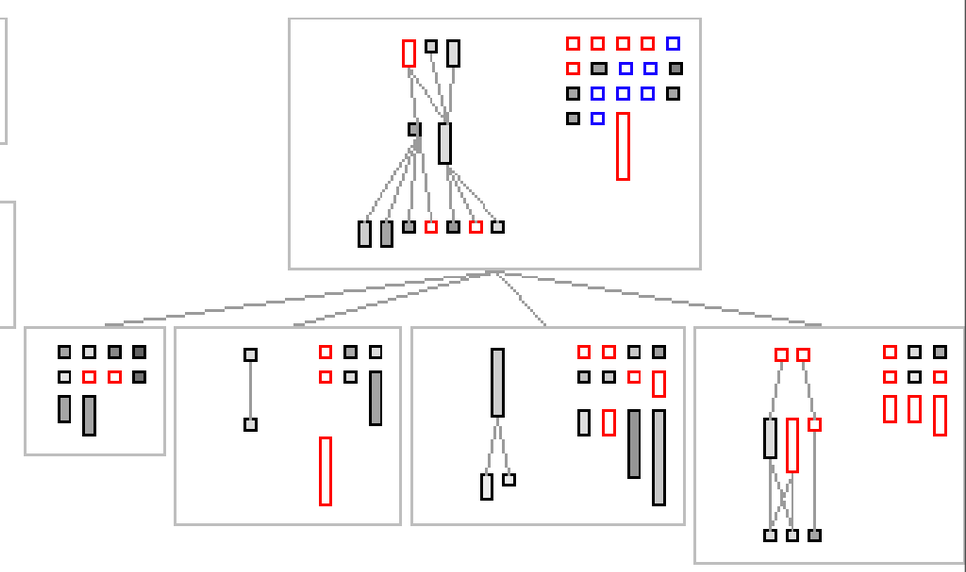
Number of classes: 138
Number of methods: 1201
Number of tested methods: 493
Number of test methods: 170
% coverage: 54.11
Average methods per test: 2.9

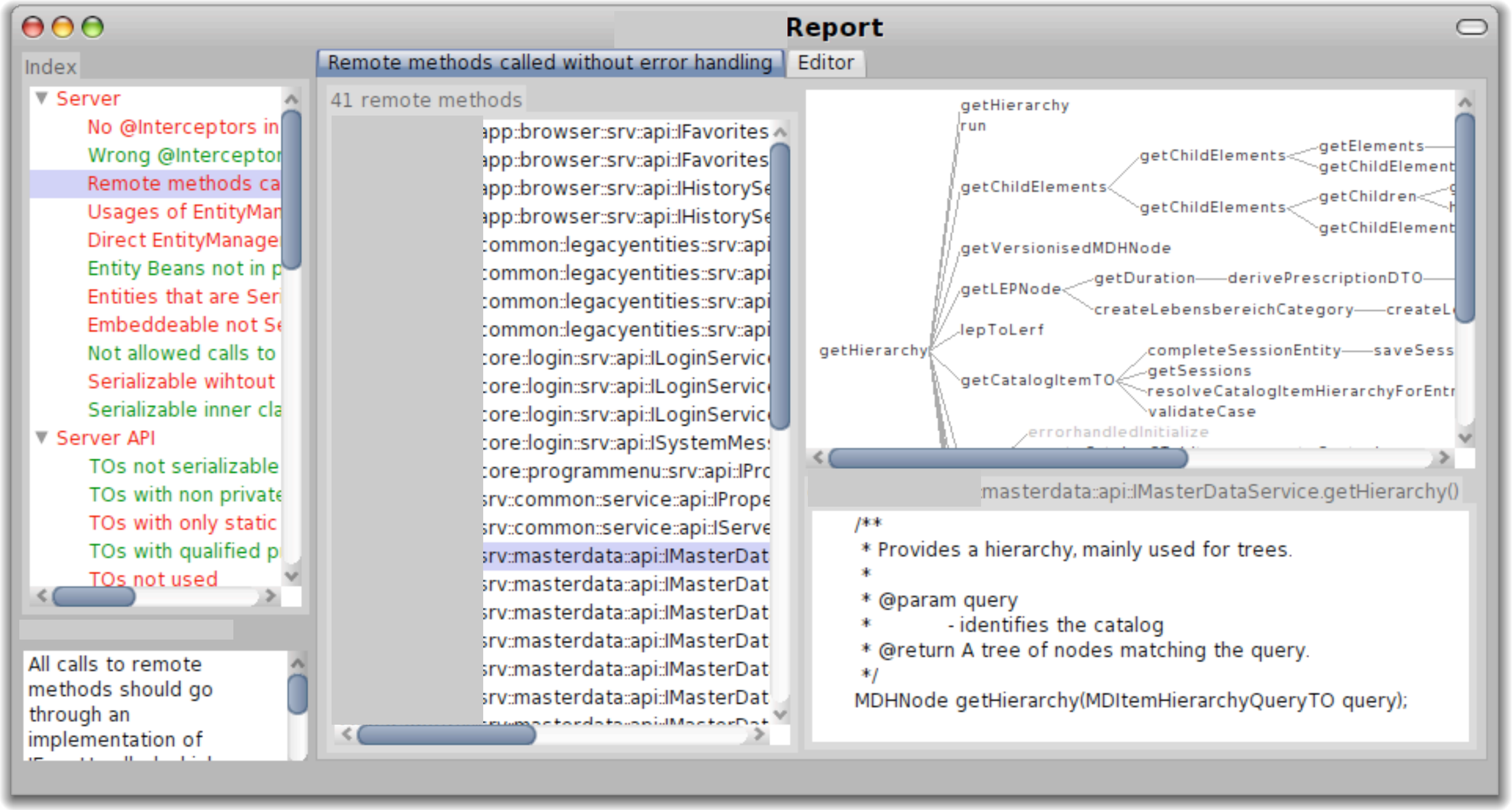












Exemple de rapports dédiés

Integration continue

-moose-report #48 Test Results [Hudson]

http://[redacted]-moose-report/lastCompletedBuild/testReport/

Hudson

search

Hudson » [redacted]-moose-report » #48 » Test Results

ENABLE AUTO REFRESH

- Back to Project
- Status
- Changes
- Console Output [raw]
- History
- Test Result
- Previous Build

Test Result

17 failures (+16)

46 tests (±0)
Took 4.5 sec.
add description

All Failed Tests

Test Name	Duration	Age
>>> Concern.(No @Interceptors in remote implementations)	0.1	1
>>> Concern.(Remote methods called without error handling)	0.1	1
>>> Concern.(Usages of EntityManager)	0.1	1
>>> Concern.(Direct EntityManager.persist() calls)	0.1	1
>>> Concern.(Entities that are Serializable)	0.1	1
>>> Concern.(Embeddeable not Serializable)	0.1	1
>>> Concern.(TOs with only static attributes)	0.1	1
>>> Concern.(TOs not used)	0.1	1
>>> Concern.(Remote interfaces incorrectly packaged)	0.1	1
>>> Concern.(Entity Beans incorrectly packaged)	0.1	1
>>> Concern.(TOs incorrectly named)	0.1	1
>>> Concern.(Jobs scheduled without progressbar)	0.1	1
>>> Concern.(Usages of StringBuffer)	0.1	1
>>> Concern.(Usages of java::util::Vector)	0.1	1
>>> Concern.(Usages of setTimeInMillis)	0.1	1
>>> Concern.(Attributes not following naming convention)	0.1	1
>>> Concern.(Serializable wihtout default constructor)	0.1	15

Intéressé par vos problèmes

- extraction de règles
- cartographie
- prédiction de coût
- impact change
- languages propriétaires
- extraction d’architecture orientée service
- outils d’analyses

L'offre *Synectique*

Des outils dédiés, adaptés et orientés solutions

Bénéfiques au niveau coût

http://www.synectique.eu