

Language Independent Metric Support towards Refactoring Inference

Yania Crespo¹, Carlos López², Esperanza Manso¹, Raúl Marticorena²

¹University of Valladolid, Spain
{yania, manso}@infor.uva.es

²University of Burgos, Spain
{clopezno, rmartico}@ubu.es



QAOOSE 2005

9th Workshop on Quantitative Approaches in
Object-Oriented Software Engineering

ecoop
2005
European Conference
on Object-Oriented Programming

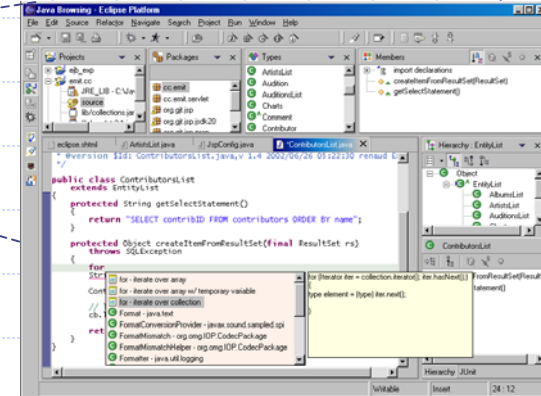
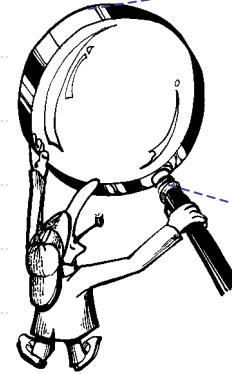
Outline

- ◆ Initial Context
- ◆ State of the Art and Current Trends
- ◆ Bad Smell and Metric Relations
- ◆ Support Based on Frameworks
- ◆ Conclusions and Future Works

Initial Context

◆ Key subject

- *When and where refactor?*



- Symptoms / stinks → Bad Smells
 - *"certain structures in the code that suggest the possibility of refactoring"* [Fowler, 2000]
 - *detection achieved from "the programmer intuition and experience"*

Initial Context

- ◆ Large number of IDEs include
 - refactoring capabilities
 - ◆ E.g: Eclipse, NetBeans, Visual Studio .NET
 - metrics plug-ins / tools
 - ◆ E.g: Metrics (Eclipse), JDepend, NDepend
- ◆ Problems

Metrics vs. Refactoring

Initial Context

◆ Two main points:

- use metrics as clues of bad smells
- define a language independent metric collection support
 - ◆ using frameworks
 - ◆ aims:
 - reuse in IDEs
 - multi-language environment
 - ◆ in accord with a current trend on language independent refactoring

State of the Art and Current Trends

◆ Bad smells

- Defined 22 in Fowler's book [Fowler, 2000]
 - ◆ each bad smell is associated to a set of refactorings
- Taxonomies [Mäntylä, 2004]
 - ◆ Bloaters
 - ◆ Object-Oriented Abusers
 - ◆ Change Preventers
 - ◆ Dispensables
 - ◆ Encapsulators
 - ◆ Couplers
 - ◆ Others

■ Current Problem

- ◆ Subjective relation between metrics and bad smells

State of the Art and Current Trends

◆ Other proposals

- change metrics used among different versions to detect which refactorings (and where) have been applied [Demeyer et al., 2000] [Gîrba et al., 2004]
- heuristics to detect refactorings opportunities
- logic meta-programming environment [Tourwé et al., 2003] [Muñoz, 2003]

◆ Other trends

- collect metrics using information available in a metamodel

Bad Smell and Metric Relations

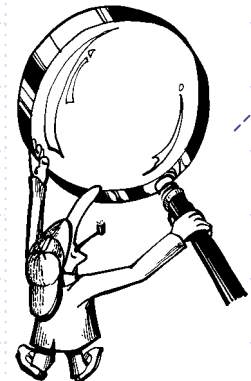
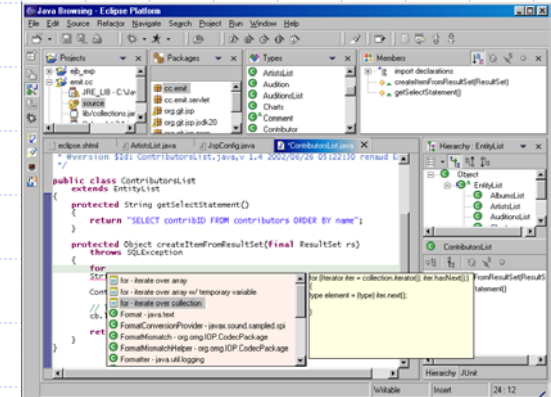
(I) A case study

◆ Definition

- Subject → JFreeChart (1.0.0_pre2)
 - ◆ class library for generating charts in Java
 - ◆ more than 600 classes
 - ◆ more than 5.000 methods
 - ◆ more than 10.000 lines of code

◆ Questions

- ◆ Where do we begin to refactor?
- ◆ When do we begin to refactor?



Bad Smell and Metric Relations

(II) A case study

◆ Scope

- Use widely accepted metrics [Chidamber & Kemerer, 1994] [McCabe 1976] [Lorentz & Kidd, 1994]
 - ◆ Eclipse + Metrics plug-in
- Selected / analyzed:

| Category | Bad Smell |
|-------------------------|--------------------------------|
| Dispensables | Data Class |
| | Lazy Class |
| Object Oriented Abusers | Switch Statements |
| Change Preventers | Parallel Inheritance Hierarchy |

Bad Smell and Metric Relations

(I) Results

◆ Data Class

- *“There are classes that have fields, getting and setting methods for fields and nothing else”*
- Metrics: NOA, NOM, WMC, LCOM
- Detected:
 - ◆ AbstractRenderer, ChartPanel, PiePlot, XYPlot and CategoryPlot
- Refactorings to be applied [Fowler, 2000]: *Move Method* to add more functionality to these classes, *Encapsulate Field* and *Encapsulate Collection*

Bad Smell and Metric Relations

(II) Results

◆ Lazy Class

- “A class that isn’t doing enough to pay for itself should be eliminated”
- Metrics: NOA, NOM, WMC, DIT
- Detected:
 - ◆ CountourPlotUtilities, DataSetReader, ChartFactory (DIT=1)
 - ◆ DefaultKeyedValues2DDataset, DefaultKeyedValuesDataSet
- Refactorings to be applied [Fowler, 2000]: *Move Method, Remove Class, Collapse Hierarchy* and *Inline Class*

Bad Smell and Metric Relations

(III) Results

◆ Switch Statements

- “Most times you see a switch statement you should consider polymorphism”
- Metrics: V(G), LOC, NBD
- Detected:
 - ◆ `executeQuery` method in `JDBCXYDataSet`
- Refactorings to be applied [Fowler, 2000]: *Replace Conditionals with Polymorphism* and *Replace Type Code with Subclass* / *Replace Type Code with State/Strategy*, *Extract Method*

Bad Smell and Metric Relations

(IV) Results

◆ Parallel Inheritance Hierarchy

- *“Every time you make a subclass of one class, you also have to make a subclass of another ”*
- Metrics: **DIT, NOC**
- Detected:
 - ◆ 3 parallel hierarchies
- Refactorings to be applied_[Fowler, 2000]: Move Method and Move Field

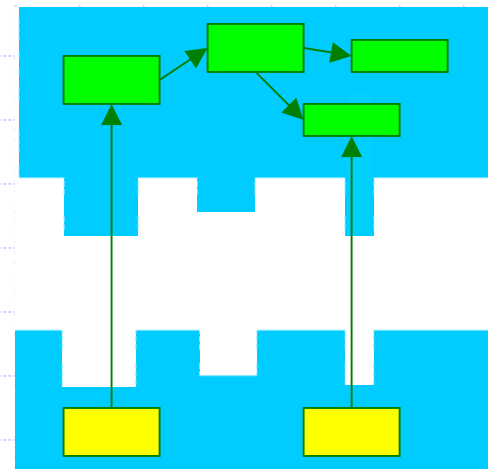
Bad Smell and Metric Relations

Conclusions

◆ Case study shows:

- Objective relation between metrics vs. bad smells (vs. refactorings)
- Relations could be established with language independent metrics
- How to seize this opportunity?
 - ◆ Give a definition based on frameworks
 - "A framework is a set of cooperating classes that make up a **reusable** design for a **specific class of software**" [Deutsch, 89]

language independence



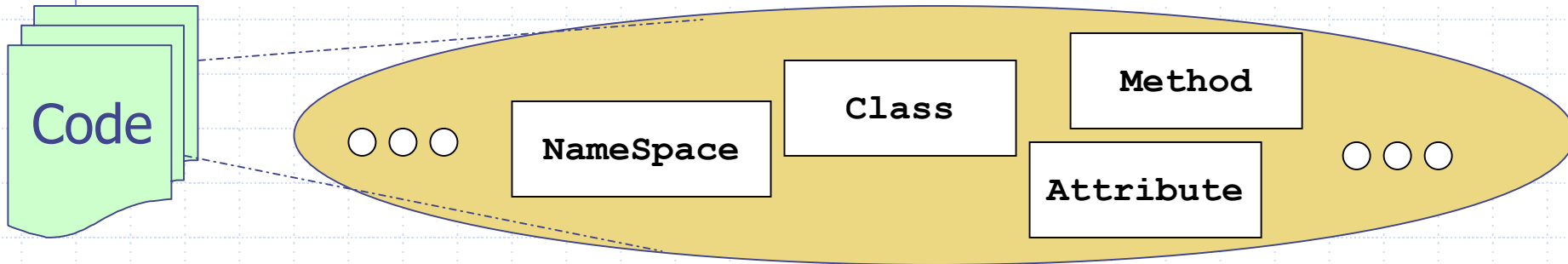
Core

Extension

Support Based on Frameworks

◆ How to collect metrics?

- Framework defined on metamodels



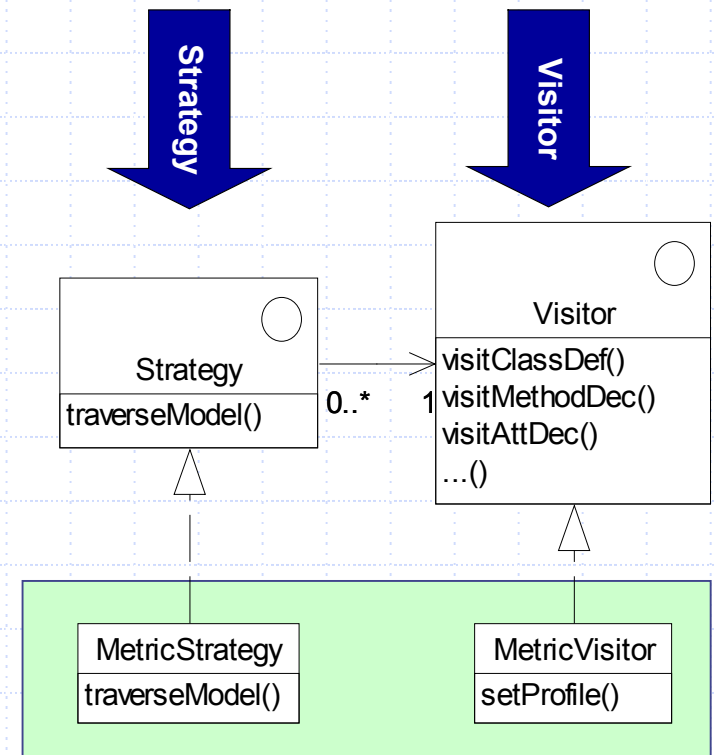
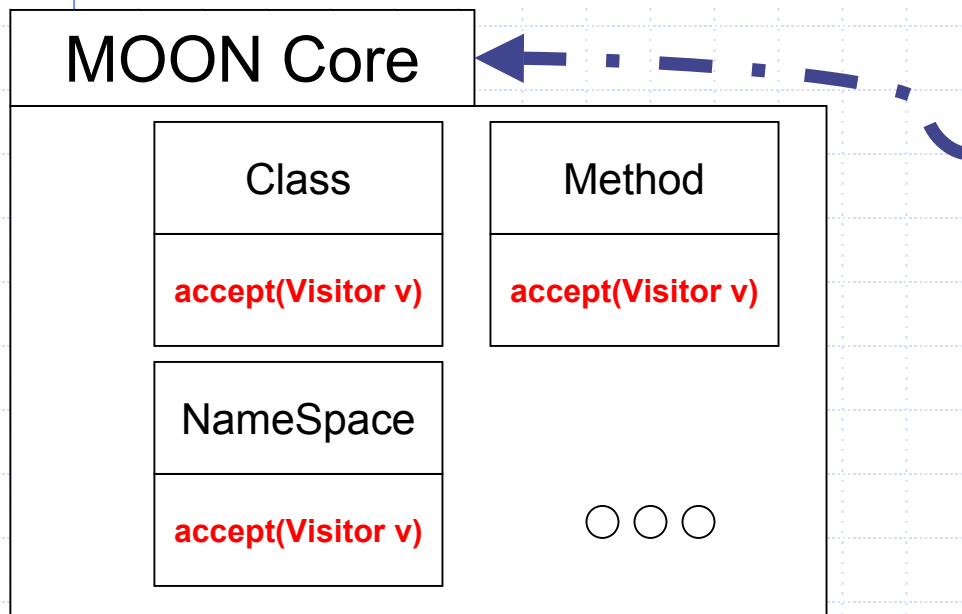
- Possible candidates
 - ◆ UML → without instructions (Actions ?)
 - ◆ FAMIX → without genericity
 - ◆ **MOON metamodel as solution**
 - **Minimal Object-Oriented Notation**

Support Based on Frameworks

Metamodel Elements Traversal

◆ Traversal of the elements

- *Visitor DP*
- *Strategy DP*

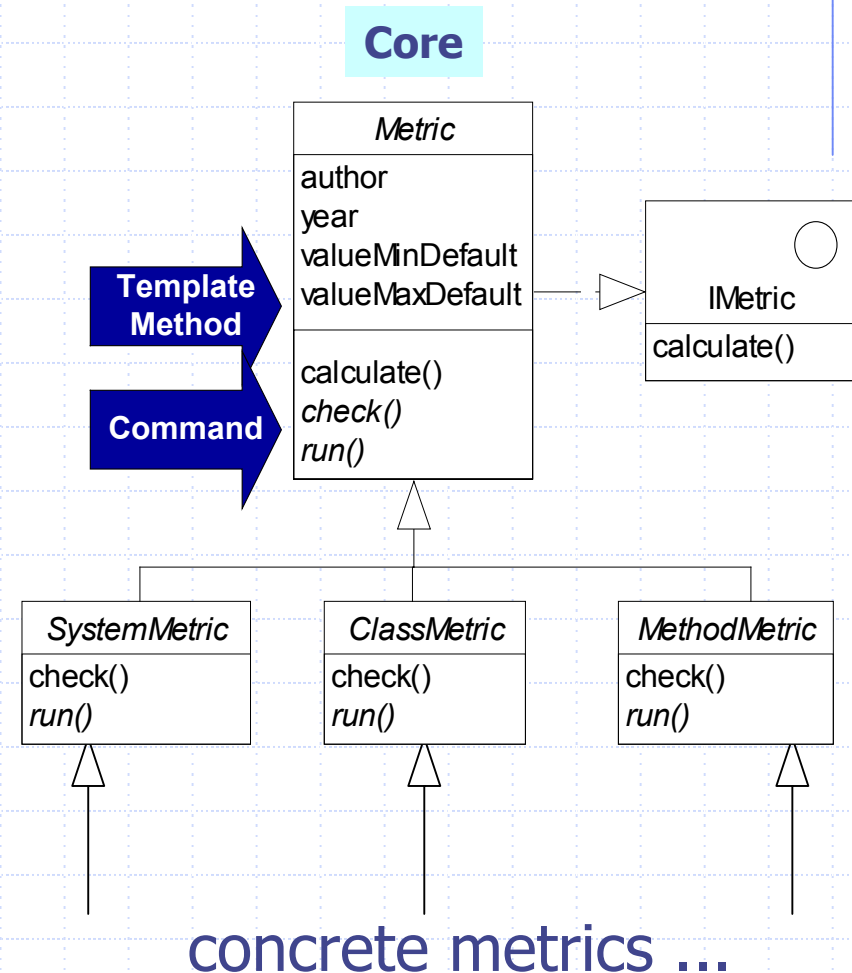


Core

Support Based on Frameworks

Runnable Metric Hierarchy

- ◆ Algorithm and elements
 - *Template Method DP*
 - general template for metrics (template method)
 - calculate
 - two phases (hook methods)
 - *check*
 - *run*
 - Different granularity of metrics: System, Class and Method
 - *Command DP*
 - concrete executions
 - run

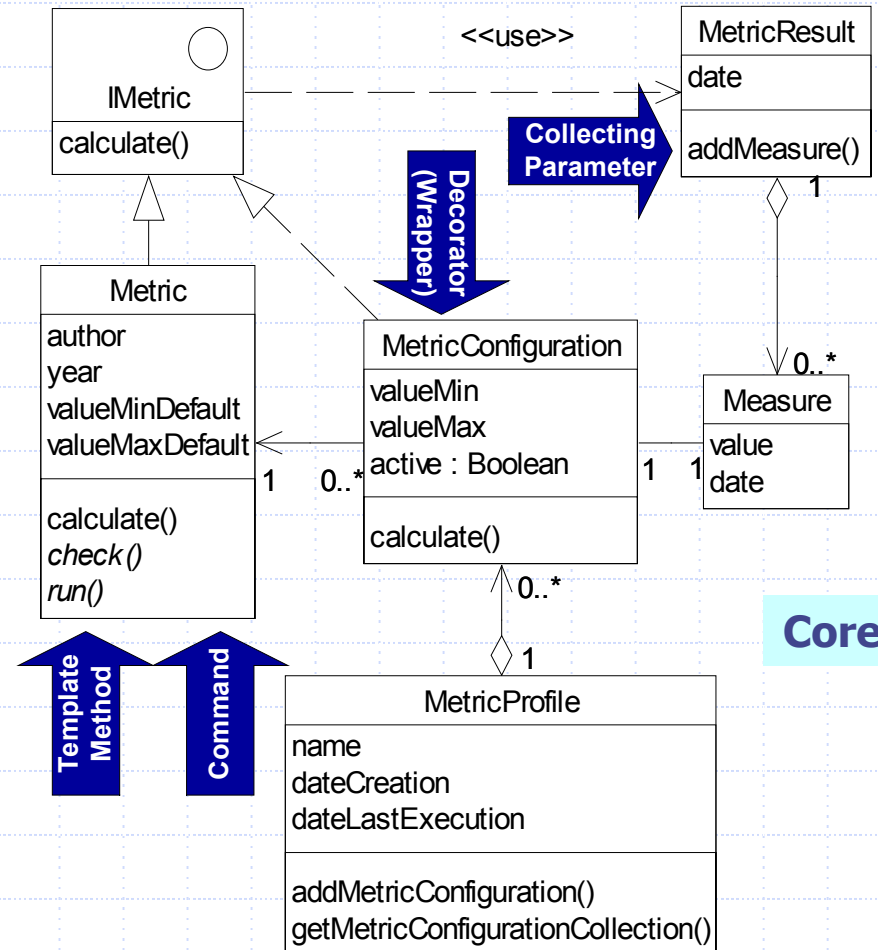


Support Based on Frameworks

Profiles: Metric Customization

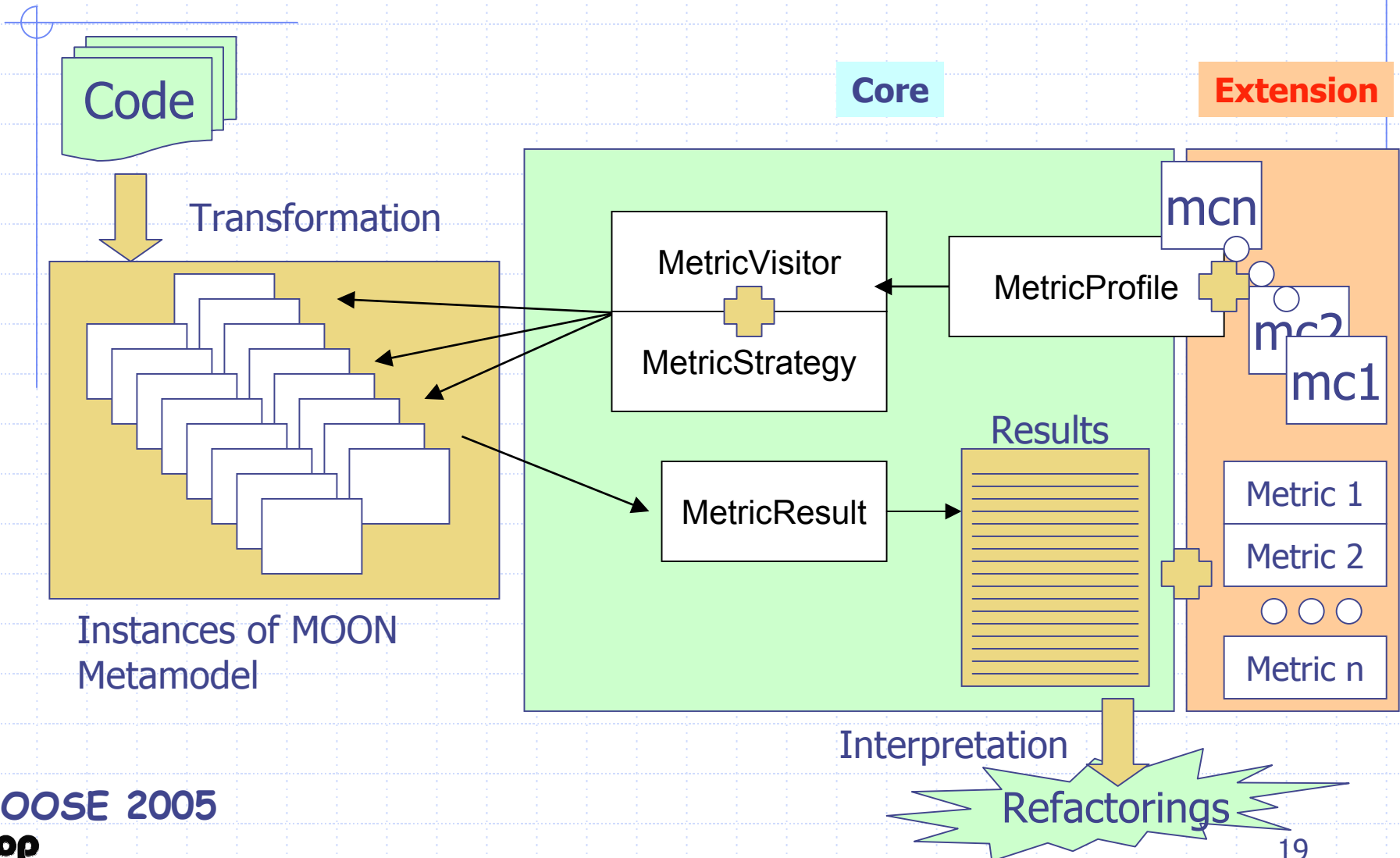
◆ Customization

- Profiles to customize each metric
- *Decorator DP*
 - Wrapper of metrics
 - Customize min and max values
- *Collecting Parameters DP*



Support Based on Frameworks

Measure Calculation



Support Based on Frameworks

Framework Validation: An Example

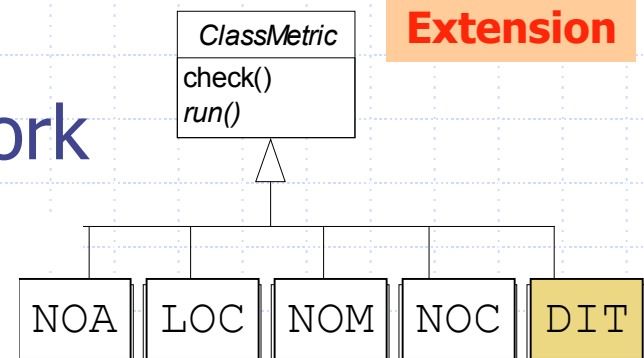
◆ Implement concrete metrics

■ Ej: DIT

- 35 lines of code (very simple)
- MOON metamodel dependence

◆ Framework provides:

- Traversal of the inheritance tree
- Collect results
- Easily plugged in framework



Strengths and Weaknesses

◆ Reuse of the framework

- Easy to include and run other metrics ...
 - ◆ language independent
 - ◆ current design developed on Java, easy migrate to other language
- Easy to change the metamodel

◆ Improvements

- include *Observer DP* to optimize calculations
- additional filters and customization of metrics
- graphical interface

Conclusions and Future Works

◆ Conclusions

- support to metric calculation
- objective method to detect refactoring opportunities

◆ Future works

- provide refactoring engines with additional module relating metrics and bad smells
- continue with empirical validation of metrics as detection way of bad smells
- face problems with certain languages

Thank you very much. Any question?

