



Soporte de Métricas con Independencia del Lenguaje para la Inferencia de Refactorizaciones

Yania Crespo¹, Carlos López² y Raúl Marticorena²

¹Universidad de Valladolid
yania@infor.uva.es

²Universidad de Burgos
{clopezno, rmartico}@ubu.es



Grupo de Investigación en
Reutilización de Objetos

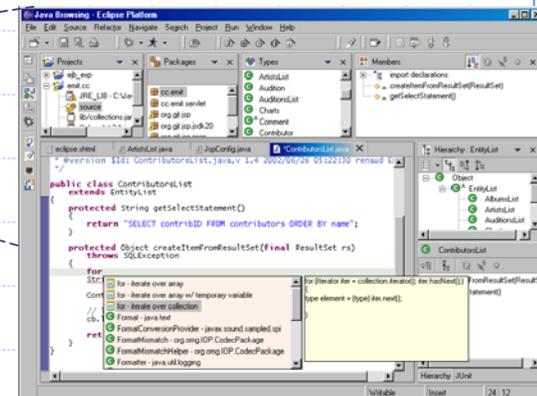
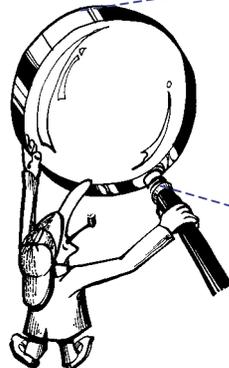
Tabla de Contenidos

- ◆ Contexto Inicial
- ◆ Estado del Arte y Tendencias Actuales
- ◆ Soporte Basado en Frameworks
- ◆ Relación entre Métricas y “*Bad Smells*”
- ◆ Conclusiones y Trabajo Futuro

Contexto Inicial

◆ Concepto clave

- *¿Cuándo y dónde refactorizar?*
- Síntomas → Bad Smells (“malos olores”)
 - *“ciertas estructuras en el código que **sugieren** la posibilidad de refactorizar” [Fowler, 2000]*
 - *detección alcanzada desde:*
 - *“la intuición y experiencia del programador”*



Contexto Inicial

- ◆ Un gran número de IDEs incluyen
 - utilidades de refactoring
 - ◆ E.g: Eclipse, NetBeans, Visual Studio .NET
 - plug-ins para el cálculo de métricas / herramientas
 - ◆ E.g: Metrics (Eclipse), JDepend, NDepend

- ◆ Problemas
 - El módulo de cálculo de métricas **se implementa para cada lenguaje**
 - ◆ **¿Por qué?**
 - La mayoría de las métricas son independientes del lenguaje
 - **¡No existe una conexión entre ambos conceptos!**
 - ◆ Métricas vs. Refactoring
 - ◆ Bad Smells como elementos de enlace

Contexto Inicial

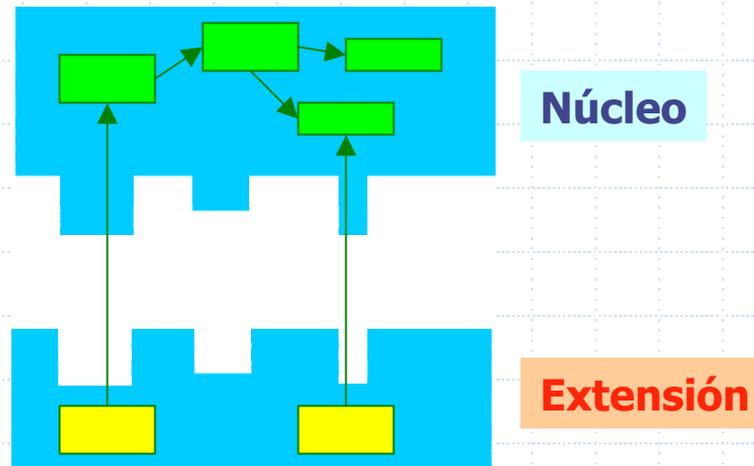
◆ Puntos principales

- definir un soporte de recolección de métricas independiente del lenguaje
 - ◆ utilizando frameworks
 - ◆ objetivos
 - reutilizar en IDEs
 - entornos multi-lenguaje
 - ◆ siguiendo una de las tendencias actuales en refactorización
- apuntar al uso de métricas como pistas /indicadores de "*bad smells*"

Contexto Inicial

- ◆ ¿Cómo aprovechar esta “oportunidad”?
 - Dando una definición basada en frameworks
 - ◆ “Un framework es un conjunto de clases cooperantes que construyen un diseño **reutilizable** para una **clase específica de software**” [Deutsch, 89]

independencia del lenguaje



Estado del Arte y Tendencias Actuales

◆ Bad smells

- Definidos 22 en el libro de Fowler [Fowler, 2000]
 - ◆ cada "bad smell" está asociado a un conjunto de refactorizaciones
- Taxonomías [Mäntylä. 2004]

◆ Oportunidades de Refactorización

- cambio de métricas entre diferentes versiones para detectar qué refactorizaciones (y dónde) han sido aplicadas [Demeyer et al., 2000] [Girba et al., 2004]
- heurísticas para detectar oportunidades de refactorización
- entornos de meta programación lógica [Tourwé et al., 2003] [Muñoz, 2003]

◆ Tendencias actuales

- recolección de métricas usando información disponible en ASTs o metamodelos

Soporte Basado en Frameworks

- ◆ ¿Cómo recolectar las métricas?
 - Framework definido sobre metamodelos



- Candidatos posibles
 - ◆ UML → sin instrucciones (¿Actions?)
 - ◆ FAMIX → sin genericidad
 - ◆ **metamodelo MOON como solución**
 - **Minimal Object-Oriented Notation**
 - “Hacia una Solución Basada en Frameworks para la Definición de Refactorizaciones con Independencia del Lenguaje”
 - **Definido un motor de refactorizaciones**
 - “Un Framework para la Reutilización de la Definición de Refactorizaciones”

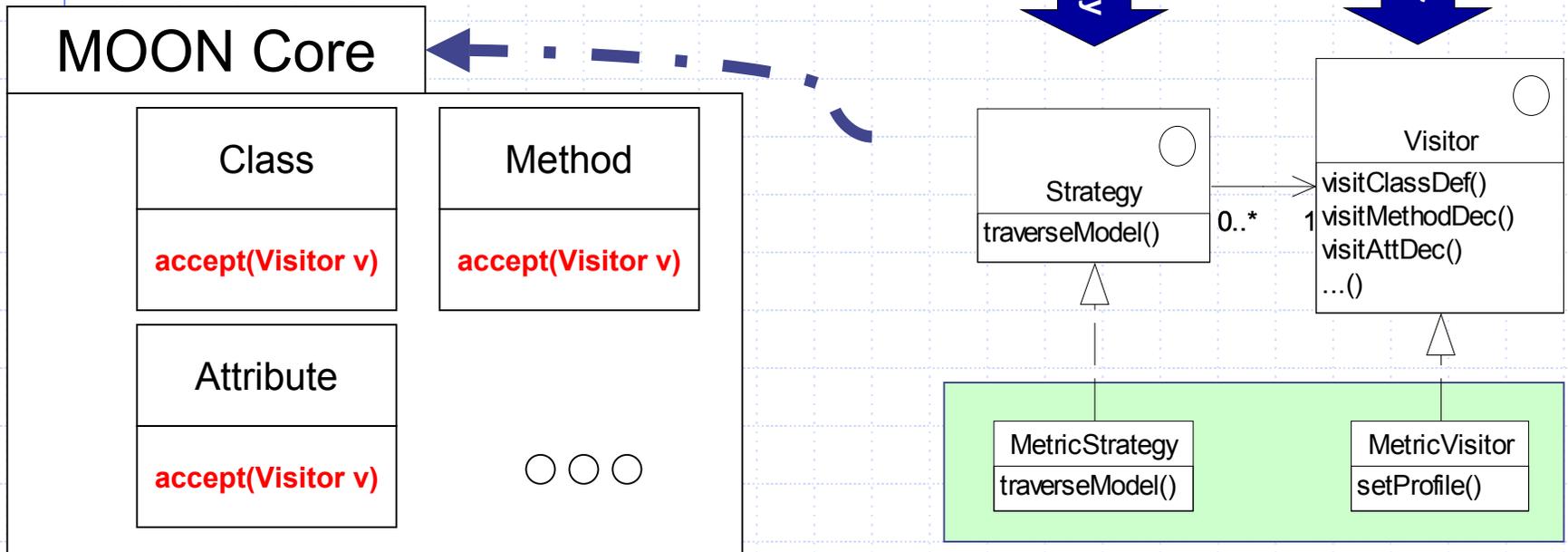


Soporte Basado en Frameworks

Recorrido de los Elementos del Metamodelo

◆ Recorrido de los elementos

- *PD Strategy*
- *PD Visitor*

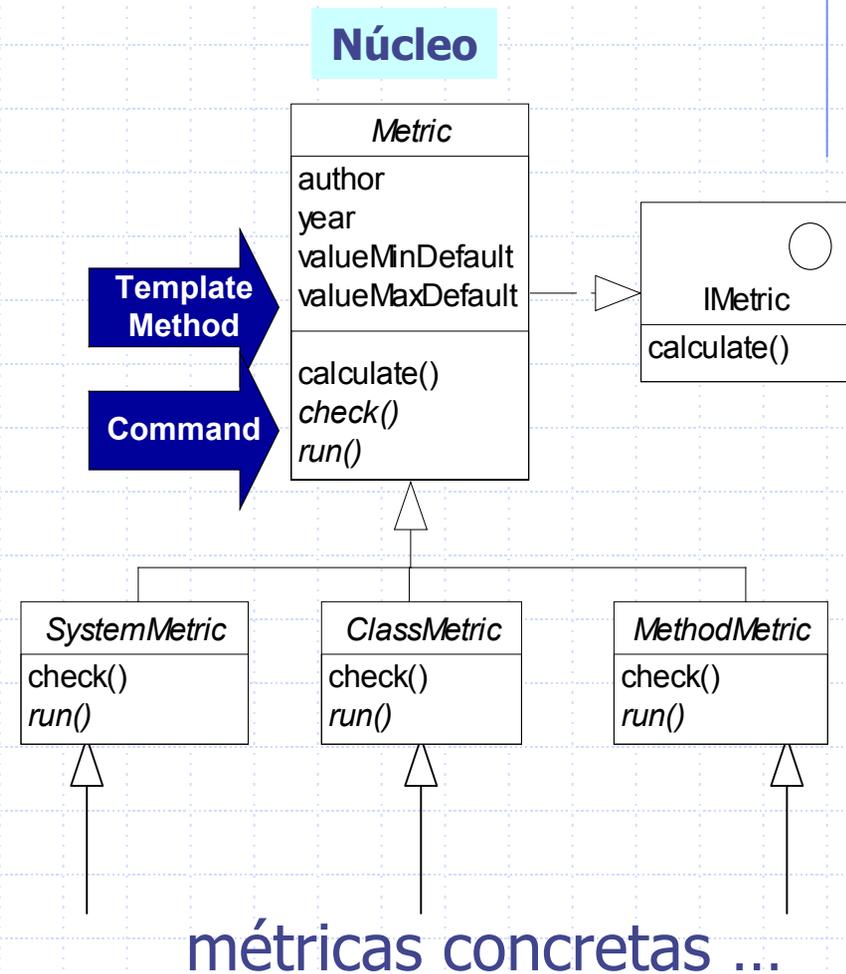


Núcleo

Soporte Basado en Frameworks

Jerarquía Ejecutable de Métricas

- ◆ Algoritmos y Elementos
 - *PD Template Method*
 - plantilla general para métricas (template method)
 - calculate
 - dos etapas (hook methods)
 - check
 - run
 - Diferentes granularidades de las métricas: System, Class y Method
 - *PD Command*
 - ejecuciones concretas
 - run



Soporte Basado en Frameworks

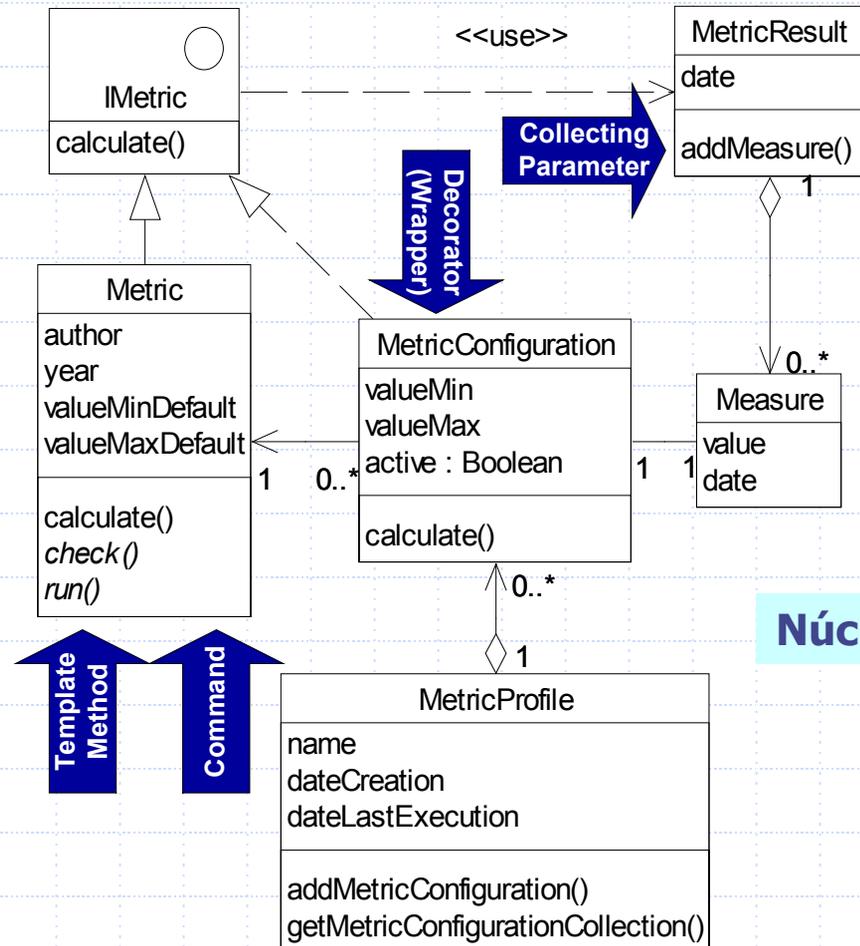
Perfiles y Resultados

◆ Personalización

- Perfiles para la personalización de cada métrica
- *PD Decorator*
 - Envuelve a la métrica
 - Personaliza valores mínimos y máximos

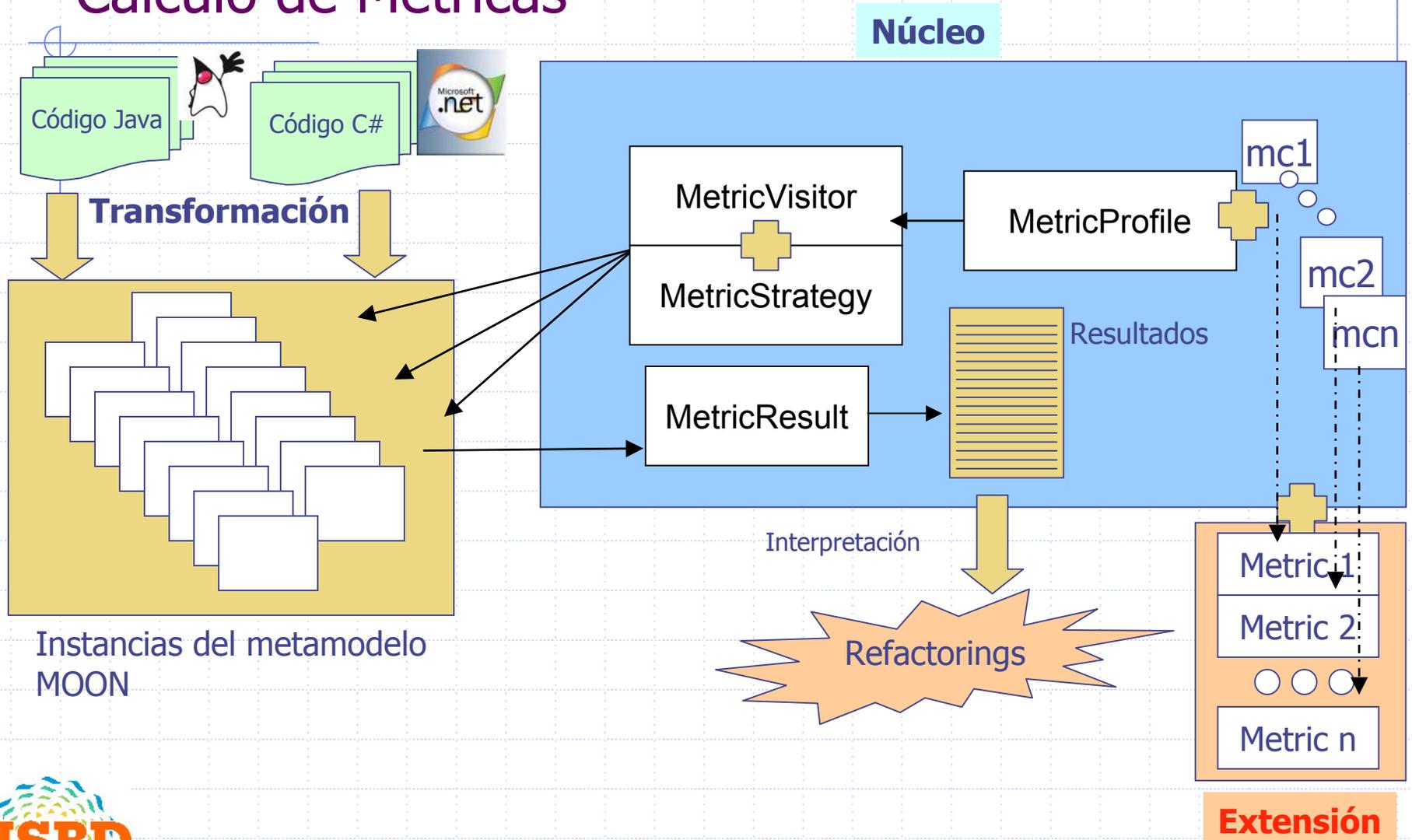
■ Resultados

- *PD Collecting Parameters*



Soporte Basado en Frameworks

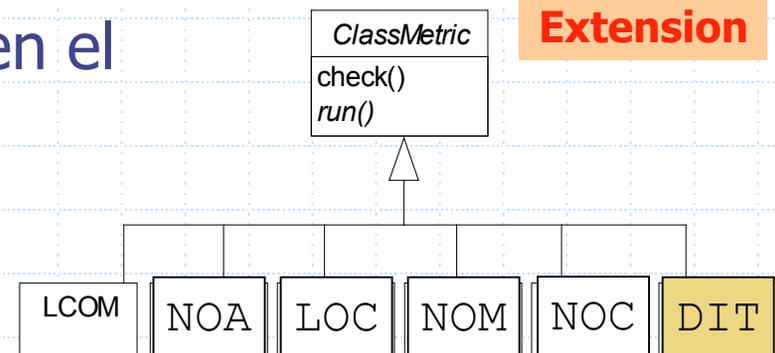
Cálculo de Métricas



Soporte Basado en Frameworks

Validación del Framework: Un ejemplo (I)

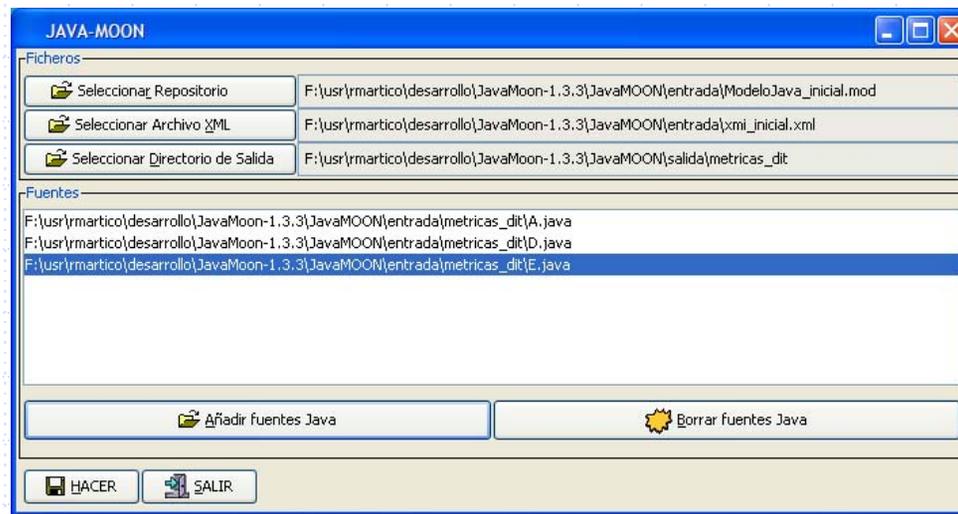
- ◆ Implementar métricas concretas
 - ◆ Ej: **DIT, 35 líneas de código (simple)**
 - ◆ dependencia (sólo) del metamodelo de MOON
 - ◆ el framework proporciona:
 - recorridos
 - recolección de resultados
 - fácilmente “conectable” en el framework



Soporte Basado en Frameworks

Validación del Framework: Un ejemplo (II)

- Fase previa: parseado de código a instancias de MOON



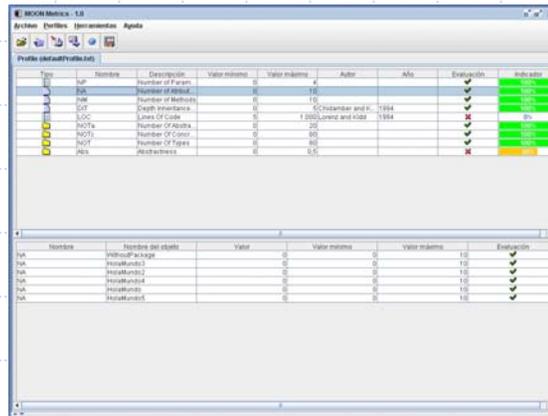
- Entradas: código fuente (java)
- Salidas: instancias MOON + extensión Java (serializables en ficheros .mod)

Soporte Basado en Frameworks

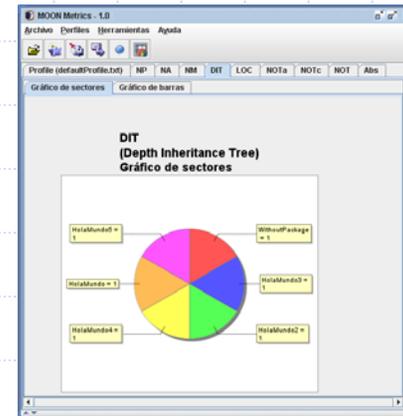
Validación del Framework: Un ejemplo (III)

■ Prototipo

- ◆ 1. Cargar el fichero con las instancias (.mod)
- ◆ 2. Cargar el perfil
 - Núcleo + Extensión del framework
- ◆ 3. Recolectar métricas
 - Núcleo del framework
- ◆ 4. Visualizar resultados

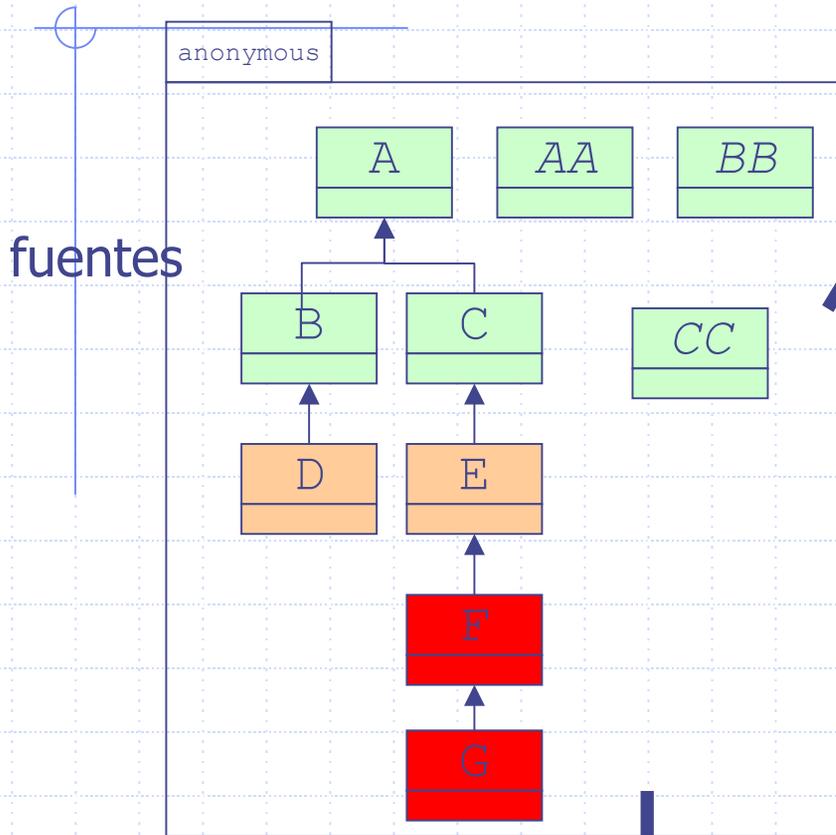


Nombre	Descripción	Valor mínimo	Valor máximo	Autor	Info	Extensión	Indicador
Fun	Number of Funct.	0	10				100%
Fun	Number of Methods	0	100				100%
Fun	Number of Methods	0	100				100%
LOC	Length Inheritance	0	5	Schuster and K...	1994		100%
LOC	Lines Of Code	0	1000	McCormack and...	1994		100%
NOIa	Number Of Abstra.	0	200				100%
NOIc	Number Of Const.	0	100				100%
NOI	Number Of Types	0	100				100%
Abs	Abstraction	0	0.5				100%



Soporte Basado en Frameworks

Validación del Framework: Un ejemplo (IV)

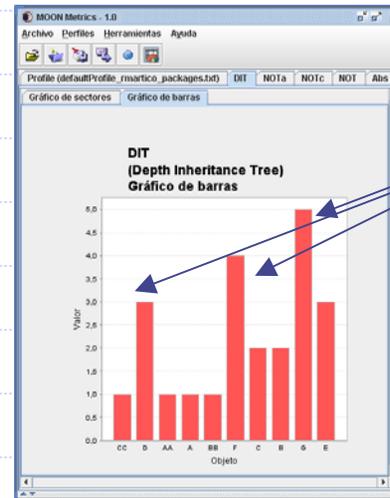


perfil

metrics.concretemetrics.DIT#Depth Inheritance Tree#Chidamber and Kemerer#1994#0.0#3.0
 metrics.concretemetrics.NOTA#Number Of Abstract Types#0.0#2.0
 metrics.concretemetrics.NOTc#Number Of Concrete Types#0.0#7.0
 metrics.concretemetrics.NOT#Number Of Types#0.0#10.0
 metrics.concretemetrics.Abs#Abstractness#0.0#0.5

MOON Metrics - 1.0

Perfil	Nombre	Descripción	Valor mínimo	Valor máximo	Autor	Año	Evaluación	Indicador
DIT	Depth Inheritance Tree		0	3	Chidamber an.	1994		
NOTa	Number Of Abstract Types		0	2				
NOTc	Number Of Concrete Types		0	7				
NOT	Number Of Types		0	10				
Abs	Abstractness		0	0.5				



síntomas

Puntos Fuertes y Débiles

◆ Reutilización del framework

- Fácil incluir y ejecutar otras métricas
 - ◆ Implementación de la "métrica" e inclusión en la biblioteca
- Independiente del lenguaje
 - ◆ con un parser del lenguaje "X" a MOON es completamente reutilizable
 - ◆ todas las métricas implementadas trabajan con instancias de MOON
- Diseño actual desarrollado en Java, fácilmente migrable a otros lenguajes
- Posibilidad de cambiar el metamodelo con un esfuerzo medio

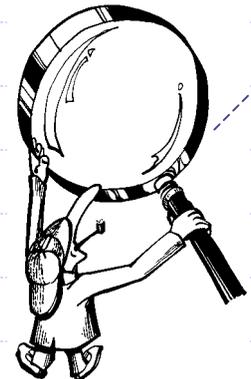
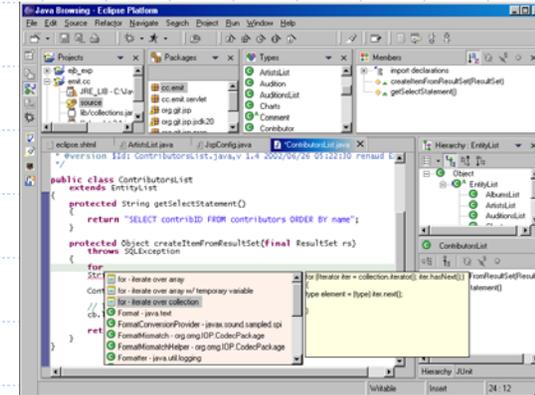
◆ Mejoras

- incluir el *PD Observer* para optimizar los cálculos
- filtros adicionales y mayor personalización de métricas
- nueva interface gráfica mejorada
- inclusión de nuevas métricas

Relación Métricas y Bad Smells (I)

Un caso de estudio

- ◆ Proceso no subjetivo definido
 - Métricas → Bad Smells → Refactoring
- ◆ Definición
 - Sujeto → JFreeChart (1.0.0_pre2)
 - ◆ biblioteca de clases para generar gráficos en Java
 - ◆ más de 600 clases
 - ◆ más de 5.000 métodos
 - ◆ más de 10.000 líneas de código
- ◆ Cuestiones
 - ◆ ¿Dónde empezamos a refactorizar?
 - ◆ ¿Cuándo empezamos a refactorizar?



Relación Métricas y Bad Smells

(II) Un caso de estudio

◆ Ámbito

- Uso de métricas ampliamente aceptadas
[Chidamber & Kemerer, 1994] [McCabe 1976] [Lorentz & Kidd, 1994]
 - ◆ Eclipse + Metrics (plug-in)
- Seleccionados /analizados:

Categoría	Bad Smell
Dispensables	Data Class
	Lazy Class

Relación Métricas y Bad Smells (I)

Resultados

Nombre	Data Class ("Clase de Datos")
Descripción	"Hay clases que tienen atributos, métodos <code>get</code> y <code>set</code> para los atributos, y nada más"
Refactorizaciones a aplicar [Fowler, 2000]	<i>Move Method</i> (para añadir más funcionalidad a estas clases) <i>Encapsulate Field</i> <i>Encapsulate Collection</i>
Métricas	NOA, NOM, WMC, LCOM
Clases Detectadas	<code>AbstractRenderer</code> <code>ChartPanel</code> <code>PiePlot</code> <code>XYPlot</code> <code>CategoryPlot</code>

Relación Métricas y Bad Smells

(II) Resultados

Nombre	Lazy Class ("Clase Vaga")
Descripción	<i>"Una clase que no está haciendo lo suficiente debería ser eliminada"</i>
Refactorizaciones a aplicar [Fowler, 2000]	<i>Move Method</i> <i>Remove Class</i> <i>Collapse Hierarchy</i> <i>Inline Class</i>
Métricas	NOA, NOM, WMC, DIT
Clases Detectadas	<code>CountourPlotUtilities</code> <code>DataSetReader</code> <code>ChartFactory</code> <code>DefaultKeyedValues2DDataset</code> <code>DefaultKeyedValuesDataSet</code>

Conclusiones y Trabajo Futuro

◆ Conclusiones

- soporte al cálculo de métricas
 - ◆ diseño del framework
 - ◆ validación práctica de la solución
 - ◆ **iSolución reutilizable!**
- establecido proceso: métricas / bad smells / refactoring
 - ◆ las relaciones podrían ser establecidas con métricas independientes del lenguaje
 - ◆ método objetivo para detectar oportunidades de refactorización

◆ Trabajos futuros

- proveer motores de refactorización con módulo adicional relacionando métricas y *bad smells*
- continuar con la validación empírica de las métricas como modo de detección de los *bad smells*
- afrontar ciertos problemas con lenguajes particulares

¿Preguntas?

