

Ingeniería del Software

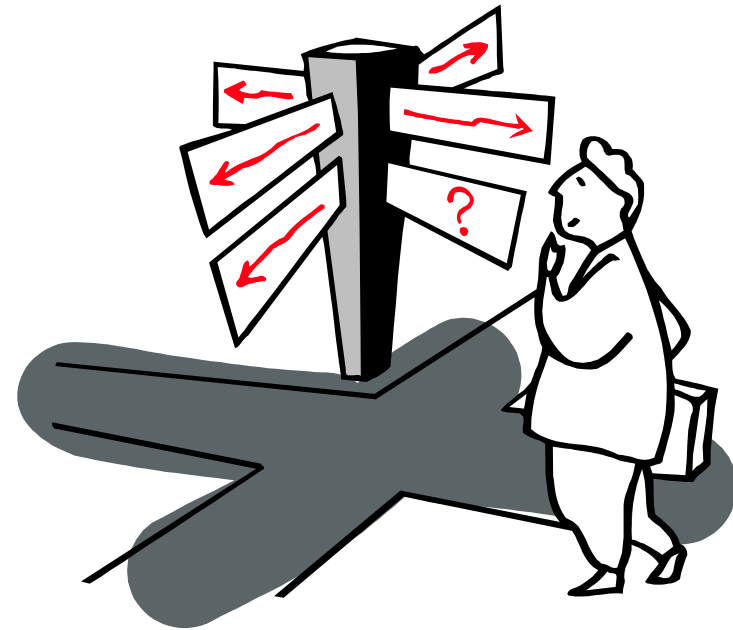


Unidad V. Métricas técnicas puestas en práctica


Gabriela Arévalo
gabriela.arevalo@lifa.info.unlp.edu.ar

Contenido

- Métricas técnicas en práctica
- Visualización 2D y Métricas
- Visualización 3D y Métricas (anécdota de la programación ágil)



Una Solución: Una Vista Polimétrica

- Una combinación “lightweight” de dos metodologías: 
 - Visualización de Software (reducir complejidad, intuitiva)
 - Métricas de Software (escalabilidad, certeza)
- Interactividad (proceso iterativo)
- No reemplaza otras técnicas, las complementa:
 - Lectura oportunística de código

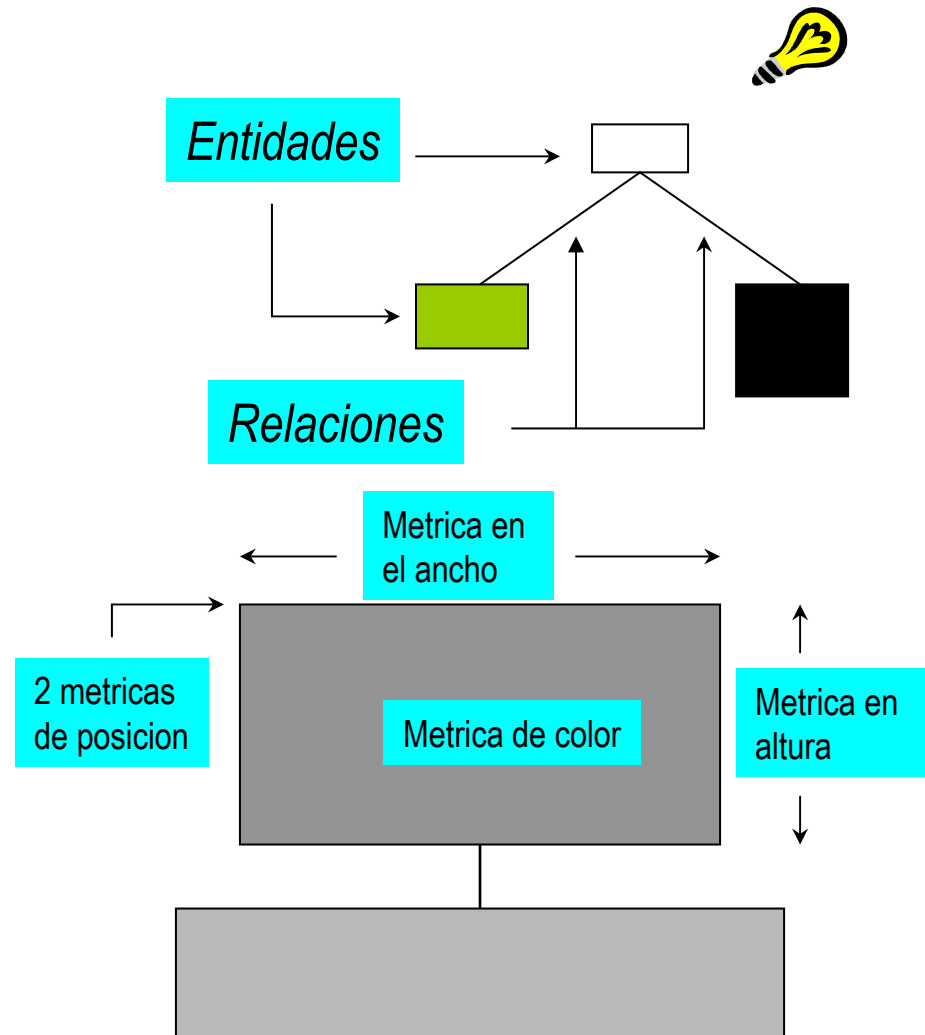
Vista Polimétrica - Principios

➤ Visualizar Software:

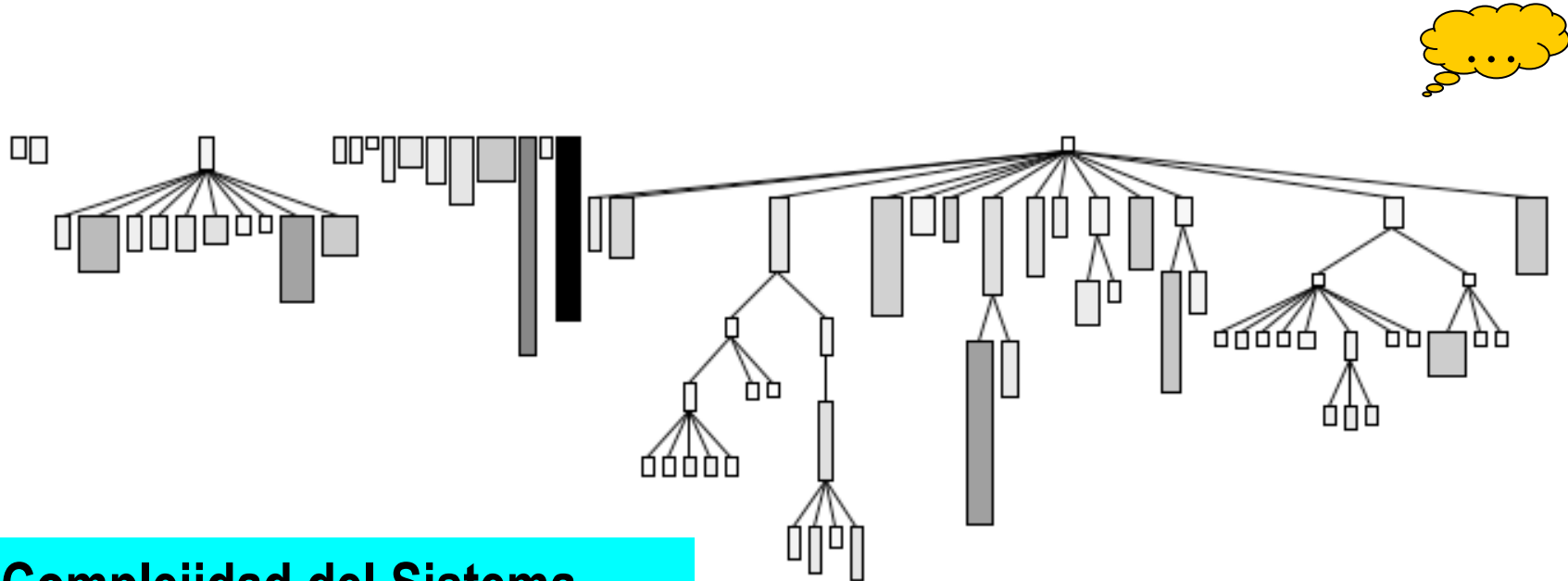
- Entidades como rectángulos
- Relaciones como aristas

➤ Enriquecer estas visualizaciones:

- Mapear hasta 5 métricas de software en una figura 2D.
- Mapear otras clases de información semántica en colores nominales.



Vista Polimétrica - Ejemplo



Complejidad del Sistema

Nodos = Clases

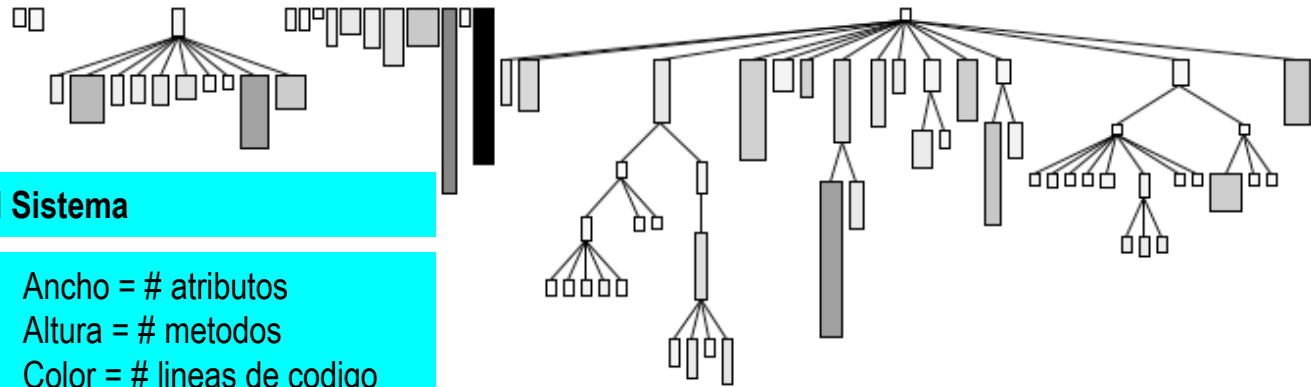
Aristas = Relaciones de herencia

Ancho = Número de Atributos

Altura = Número de Métodos

Color = Número de Líneas de Código

Vista Polimétrica - Ejemplo



Vista de Complejidad del Sistema

Nodos = Clases	Ancho = # atributos
Aristas = Relaciones de Herencia	Altura = # metodos
	Color = # lineas de codigo

Objetivos de la Ingenieria Reversa

- Obtener una impresion (construir un primer modelo mental) del sistema, conocer el tamaño, estructura y complejidad del sistema en terminos de clases y jerarquias de herencia.
- Localizar jerarquias importantes (modelo del dominio), ver si hay jerarquias profundas, o anidadas.
- Localizar grandes clases (standalone, dentro de la jerarquia de herencia), localizar clases utiles y clases con comportamiento.

Tareas soportadas por la vista

- Contar las clases, mirar los nodos desplegados, contar las jerarquias.
- Buscar jerarquias de nodos, mirar el tamaño y la forma de las jerarquias, examinar la estructura de las jerarquias.
- Buscar grandes nodos, anotar su posicion, buscar los nodos altos, buscar los nodos anchos, buscar los nodos oscuros, comparar su tamaño y forma, "leer" su nombre => lectura oportunistica de codigo

Vista Polimétrica - Descripción

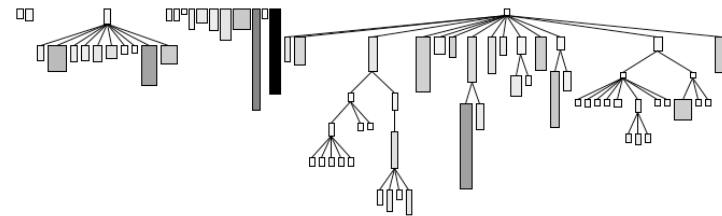
- Cada vista polimétrica se describe de acuerdo a un patrón común.
- Cada vista tiene objetivos de ingeniería reversa específicos.
- Las vistas polimétricas se implementan en CodeCrawler.



Vista de Complejidad del Sistema


Especificación Structural

Target
Alcance
Métricas
Layout
Descripción
Objetivos
Síntomas
Escenario



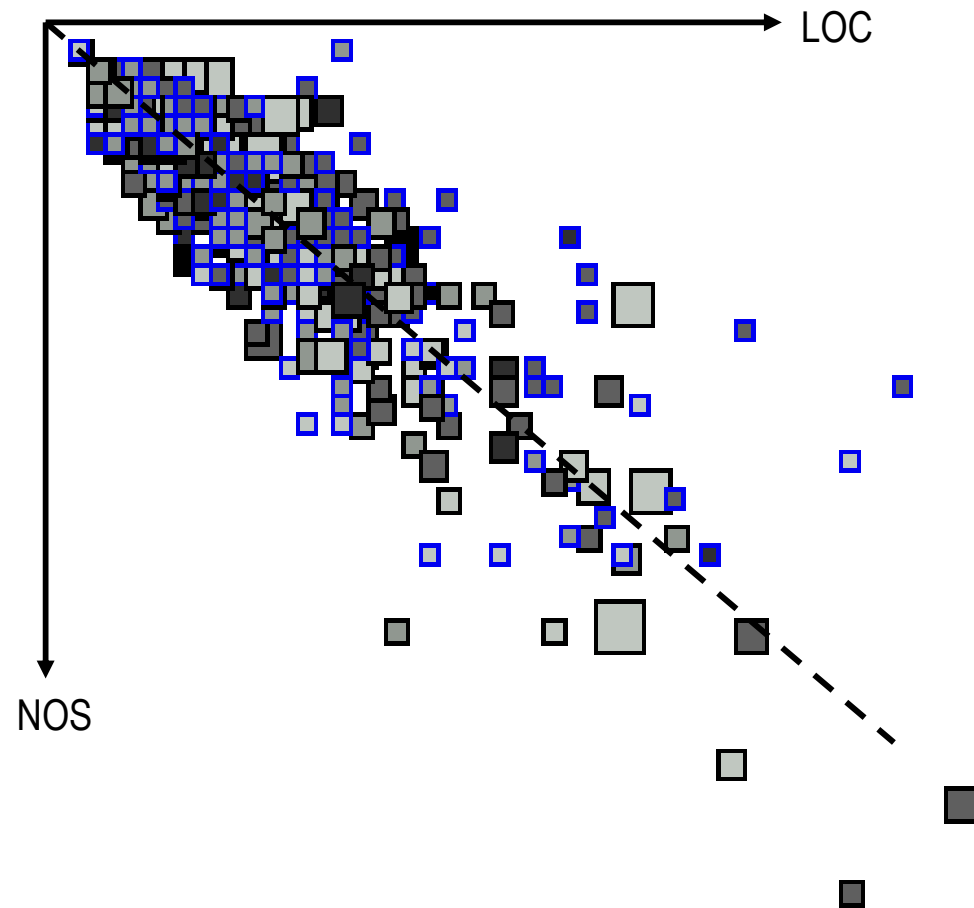
Estudio de Caso

Visualizar Software de Ampla Granularidad

- Pregunta de ingeniería reversa 
 - Cuál es el tamaño y la estructura en general del sistema?

- Objetivos de Ingeniería Reversa de Ampla Granularidad
 - Obtener un panorama global en términos de tamaño, complejidad y estructura.
 - Obtener un panorama de la calidad global del sistema
 - Localizar y entender jerarquias (del modelo del dominio) importantes.
 - Identificar grandes clases, metodos excepcionales, codigo muerto, etc.
 - ...

Vista Polimétrica de Amplia Granularidad



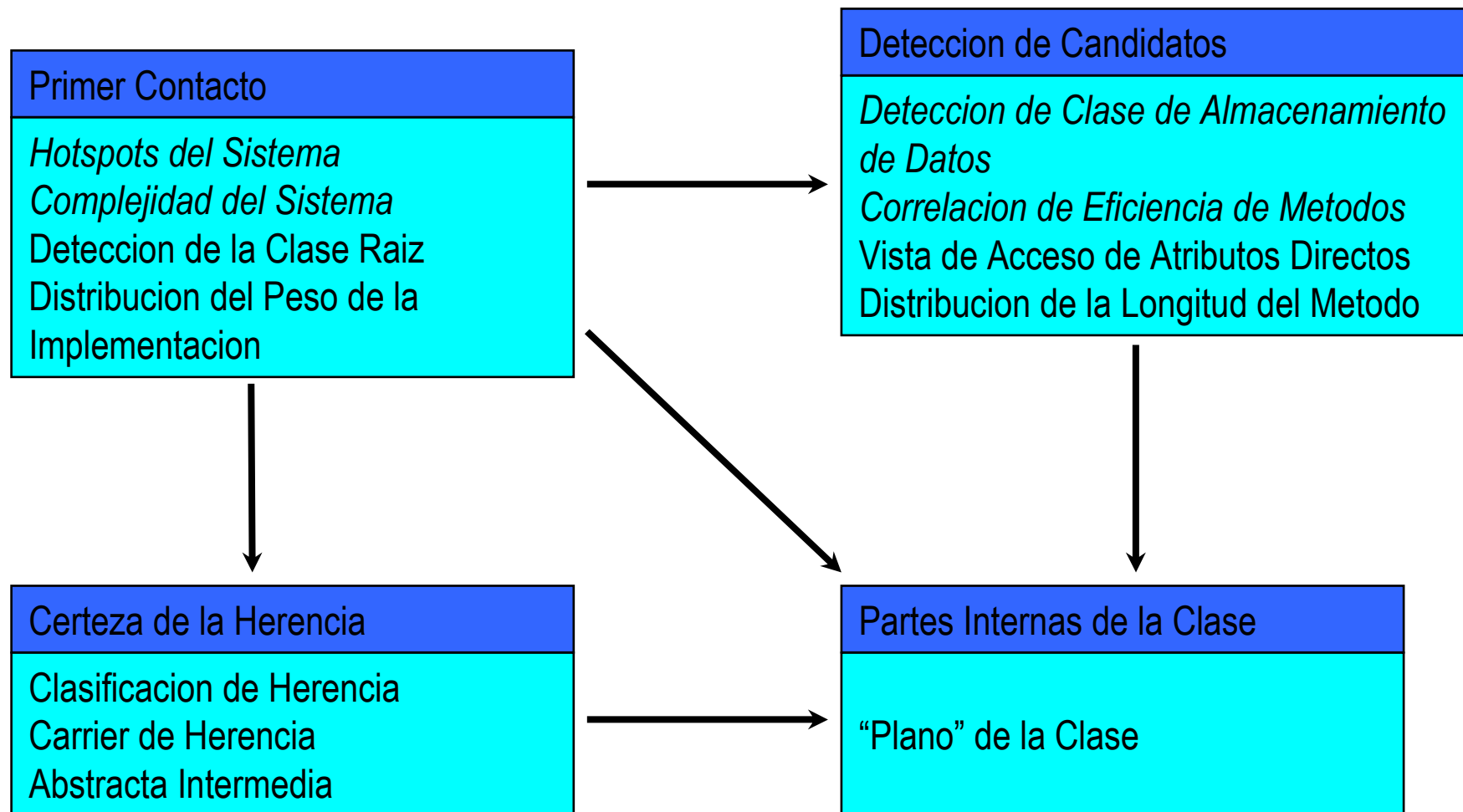
Vista de Correlacion de Eficiencia del Metodo

Nodos: Metodos
Aristas: -
Tamaño: Numero de parametros del metodo
Posición X: Number de lineas de codigo
Posición Y: Number de sentencias

Objetivos:

- Detectar metodos demasiados largos
- Detectar codigo "muerto"
- Detectar metodos incorrectamente formateados
- Conseguir una impresion del sistema en terminos del estilo de codificacion.
- Conocer el tamaño del sistema en #metodos.

Clustering las Vistas Polimétricas



Conclusiones (Amplia Granularidad)

➤ Beneficios

- Vistas son adaptables (contexto, ...) y fácilmente modificables.
- Metodologia Simple, pero poderosa
- Escalabilidad

➤ Limites

- Debe aprenderse un lenguaje visual
- Visual language must be learned

Visualización de Software de Fina Granularidad

➤ Pregunta de ingeniería reversa:

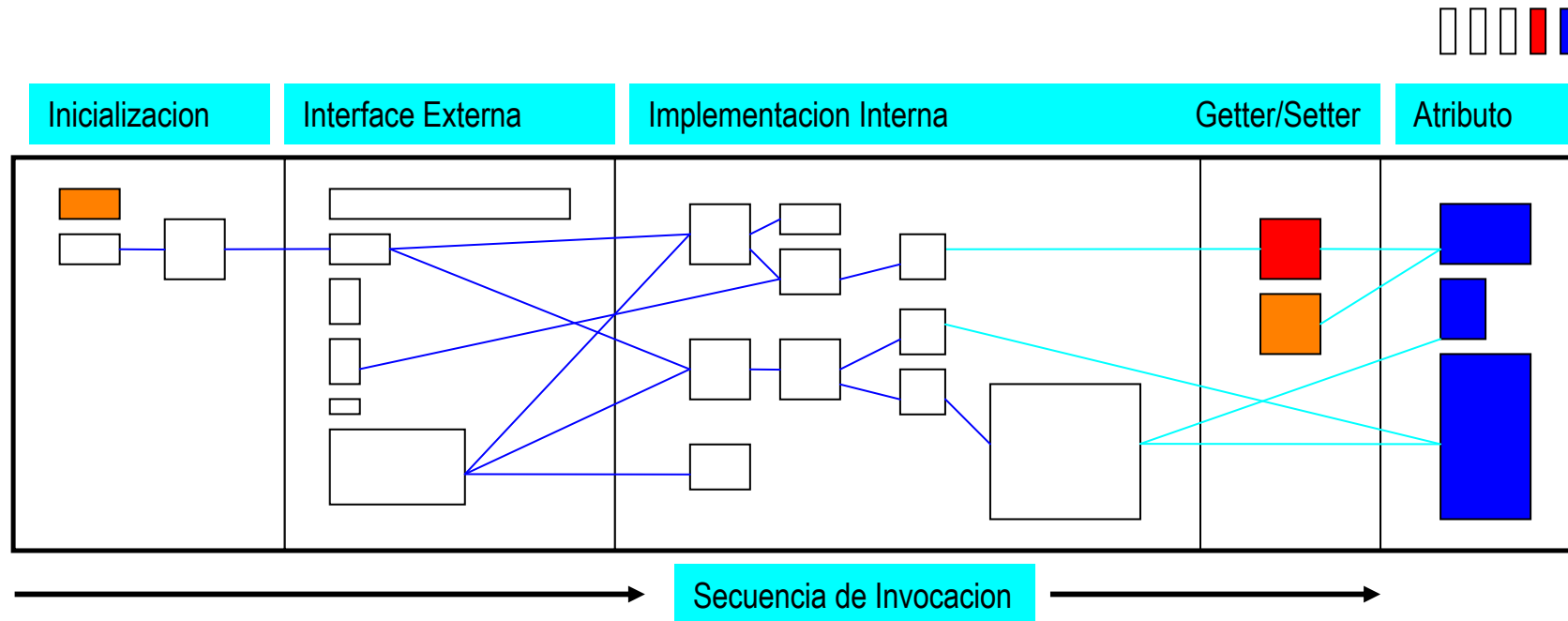


- Cual es la estructura interna del sistema y sus elementos?

➤ Los objetivos de ingeniería reversa de fina granularidad

- Entender la implementación interna de clases y jerarquía de clases
- Detectar patrones de código e inconsistencias
- Entender roles clases/subclases
- Identificar métodos claves en una clase
- ...

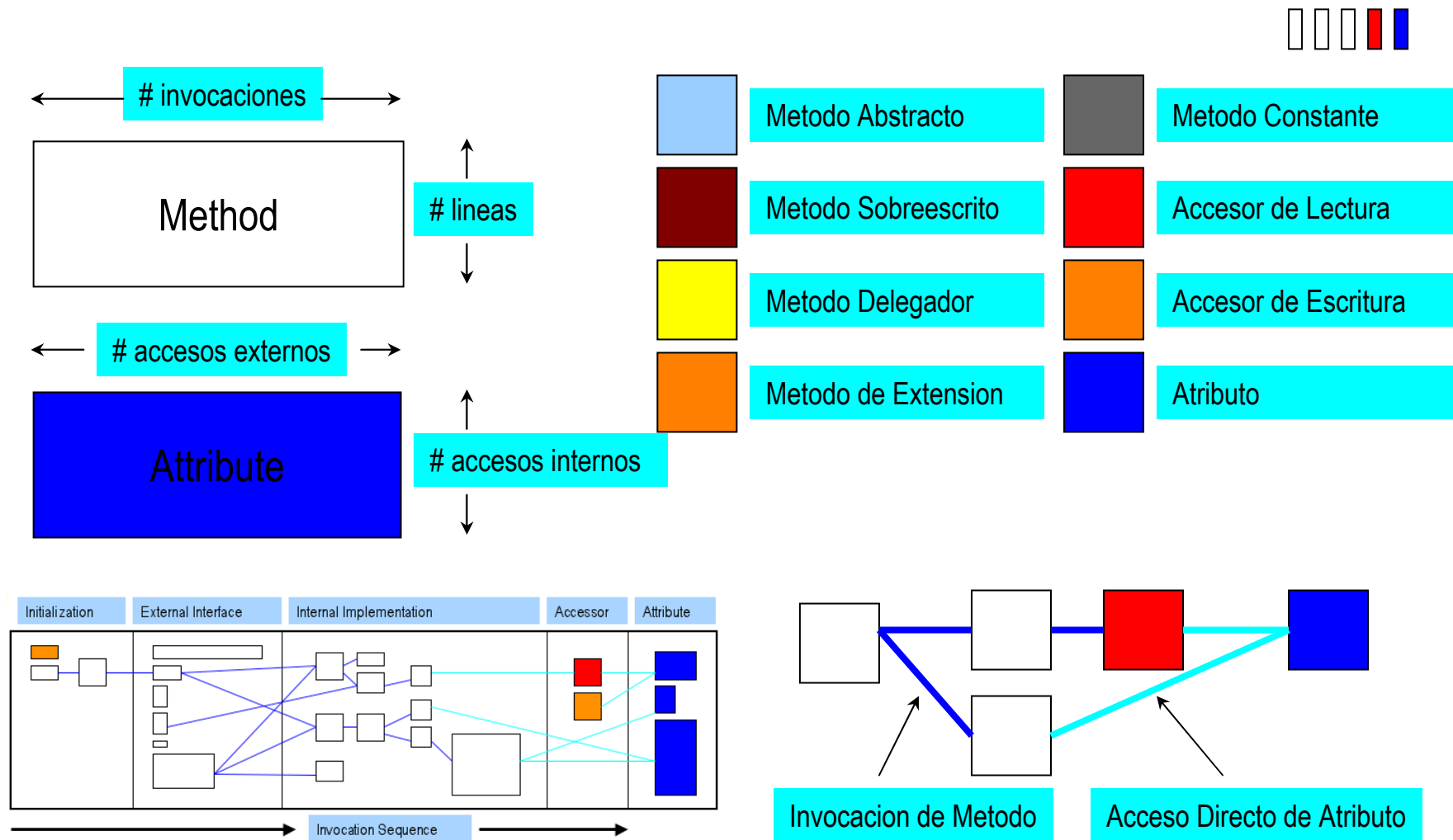
Plano de la Clase: Principios



- La clase se divide en 5 capas
- Nodos
Métodos, Atributos, Clases
- Aristas
Invocacion, Acceso, Herencia

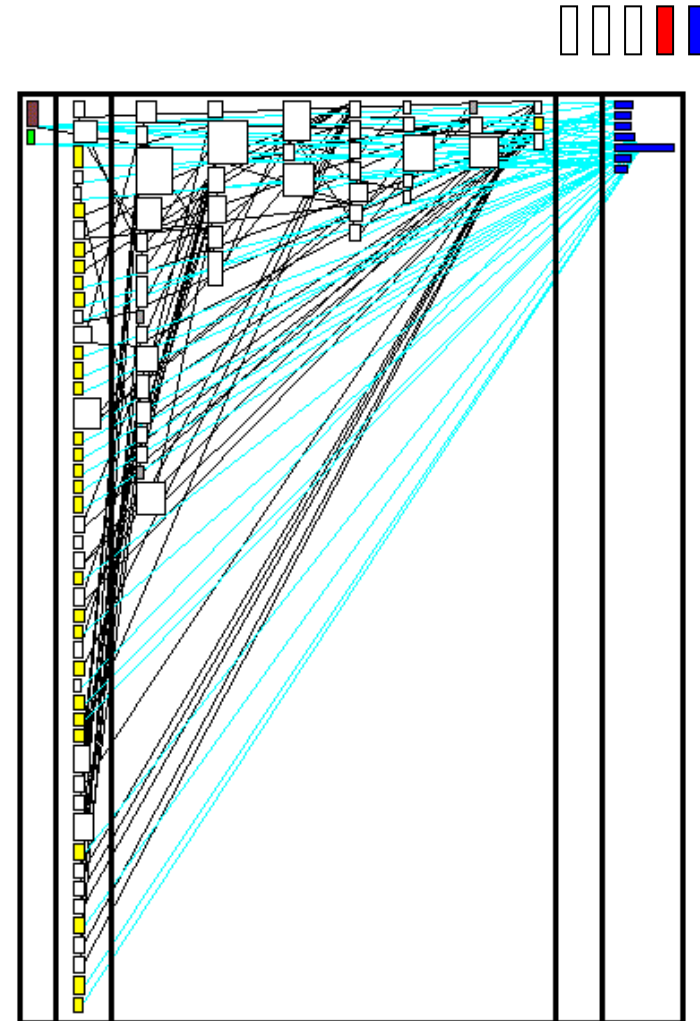
- Los nodos de metodos se posicionan de acuerdo a
La capa
La secuencia de invocacion

Plano de la Clase: Principios




Plano de la Clase - Ejemplo

- Delegacion:
 - Delega funcionalidad a otras clases
 - Puede actuar como un "Façade" (PD)
- Gran Implementacion:
 - Estructura de Invocacion Profunda
 - Varios metodos
 - Alta descomposicion
- Interface Amplia
- Acceso Directo

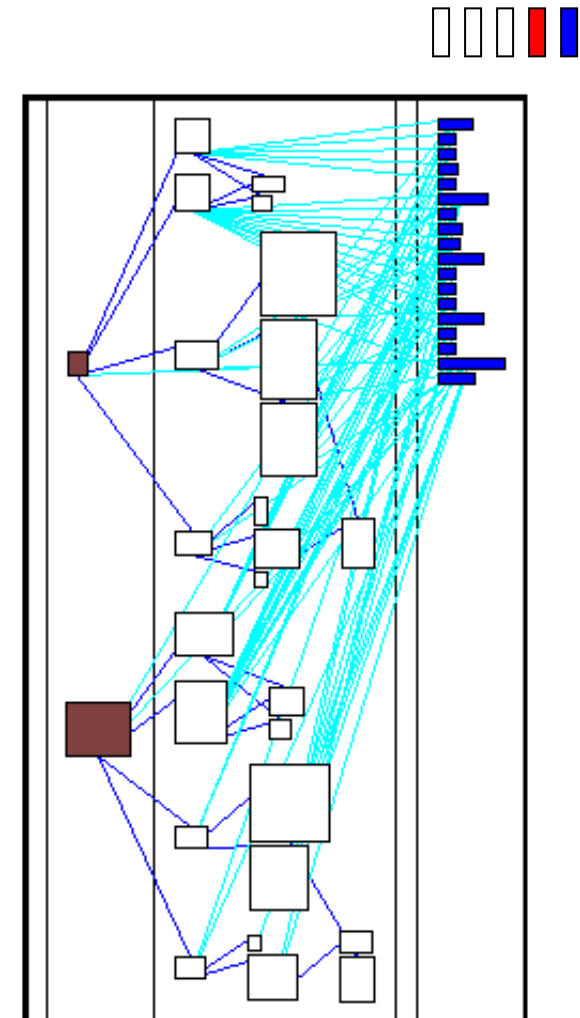


Plano de la Clase: Un lenguaje de Patrones?

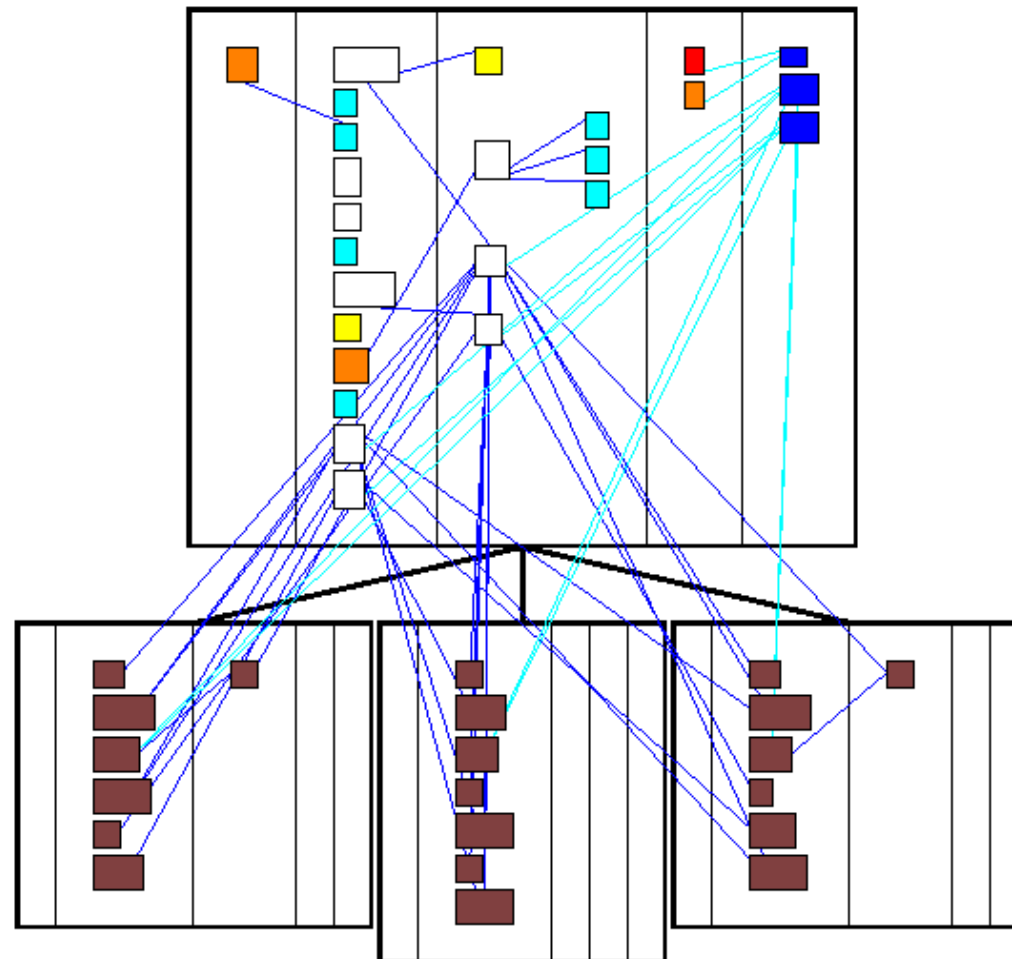
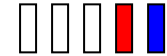
- Los patrones revelan información acerca de
 - Estilos de código
 - Políticas de código
 - Particularidades
- Se agrupan de acuerdo
 - Tamaño
 - Distribución de Capas
 - Semántica
 - Flujo de Llamado
 - Uso del estado
- Mas aun 
 - Contexto de Herencia
 - Combinaciones de patrones frecuentes
 - Combinaciones de patrones raros
- Son todos parte de un *lenguaje de patrones*

Plano de la Clase - Ejemplo

- Flujo de llamado
 - Doble entrada simple
 - (\Rightarrow separar la clase ?)
- Herencia
 - Agregada
 - Sobreescritores de Interface
- Semantica
 - Acceso directo
- Uso del Estado
 - Entradas Compartidas

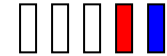


Plano de la Clase: Qué vemos?



Conclusiones (Granularidad Fina)

➤ Beneficios



- Reduccion de Complejidad
- Tecnica visual de inspeccion de codigo
- Complementa las vistas de amplia granularidad

➤ Limitaciones

- Se debe aprender un lenguaje visual
- Se requiere buen conocimiento orientado a objetos => lectura oportunistica de codigo necesaria

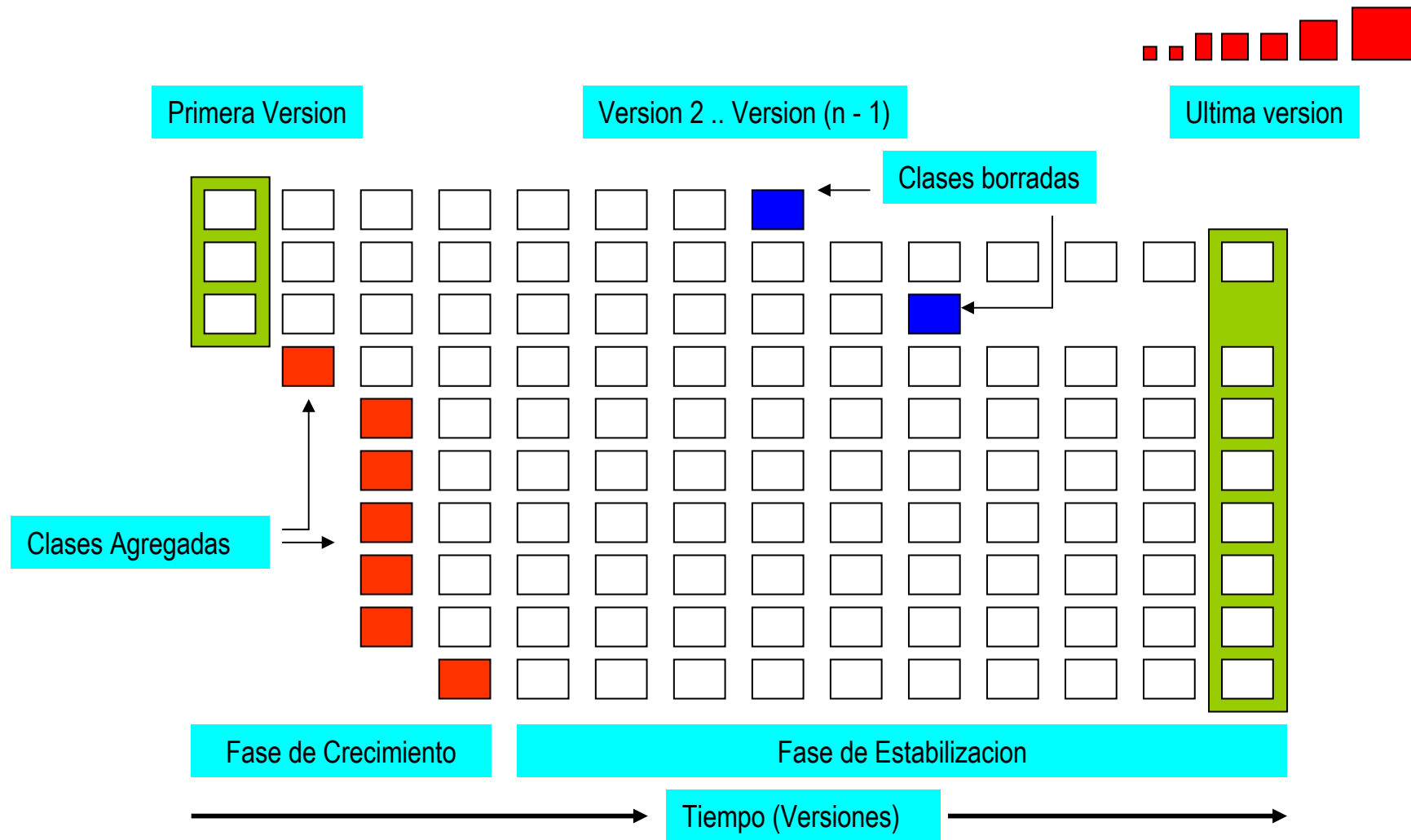
Visualizacion de Software de Evolucion

➤ Pregunta de Ingenieria Reversa:



- Cómo se llegó al sistema?
- Objetivos de Ingenieria Reversa de Evolucion
- Entender la evolucion de los sistemas orientados a objetos en terminos del tamaño e indice de crecimiento
- Entender en qué momento un elemento, p.e. una clase, ha sido agregada o borrada del sistema
- Entender la evolución de clases simples
- Detectar Patrones en la evolucion de clases
-

La Matriz de Evolución - Principios

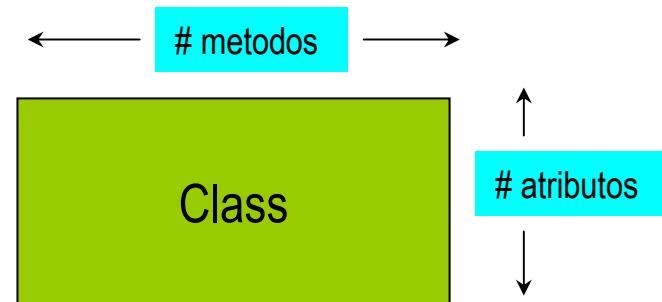
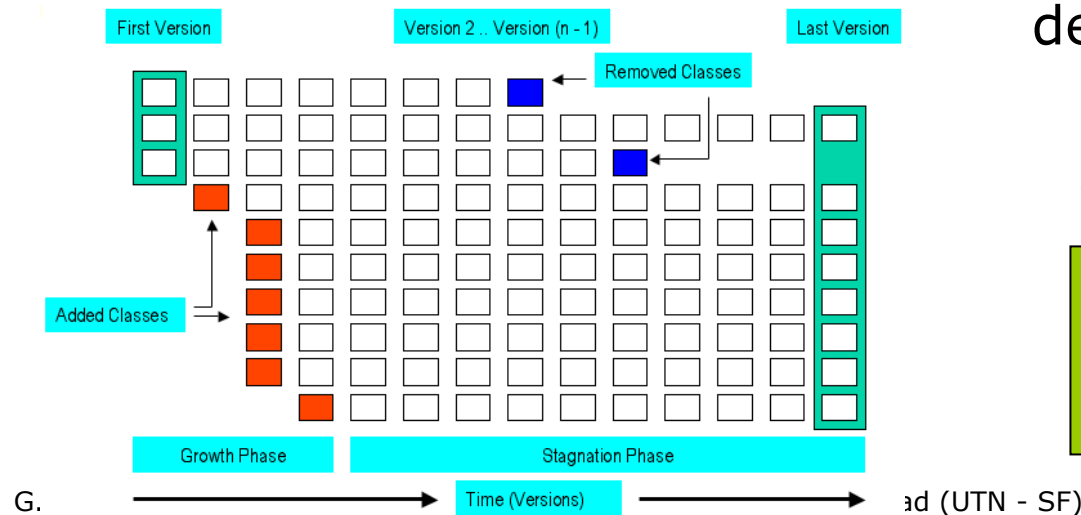


La Matriz de Evolución - Principios

- La Matriz de Evolucion revela patrones
 - La evolucion del sistema completo (versiones, crecimiento y fases de estabilizacion, indice de crecimiento, tamaños inicial y final)
 - Tiempo de vida de las clases (agregado y borrado)



- Mas aun, se enriquece la matriz de evolucion con informacion de metricas.
- Esto nos permite ver patrones en la evolucion de clases.



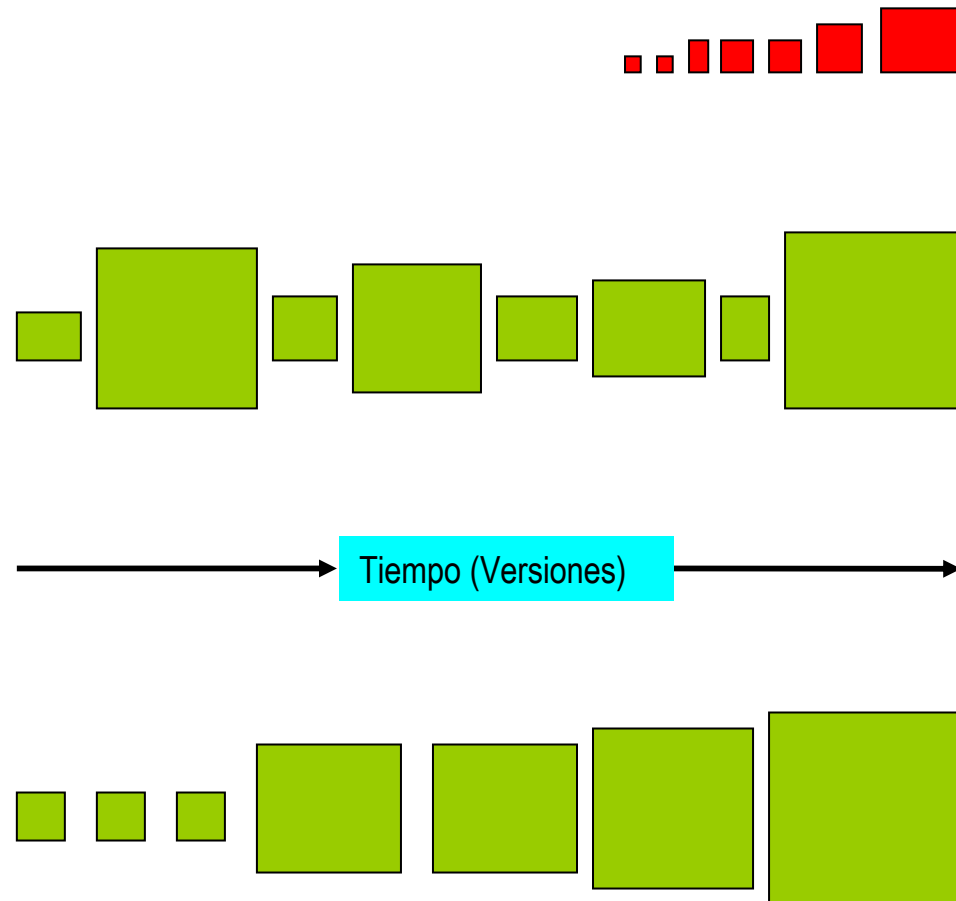
La Matriz de Evolucion: Lenguaje de Patrones

Pulsar

- Modificaciones repetidas la hacen crecer y “achicarse”
- Hotspot del Sistema: Casi en cada nueva version del sistema cambia.
- No es “clase barata”

Supernova

- De repente incrementa en tamaño, posible razones:
 - Masiva funcionalidad agregada a una clase
 - Clase de almacenamiento de datos.



La Matriz de Evolucion: Lenguaje de Patrones

“Criatura” Blanca

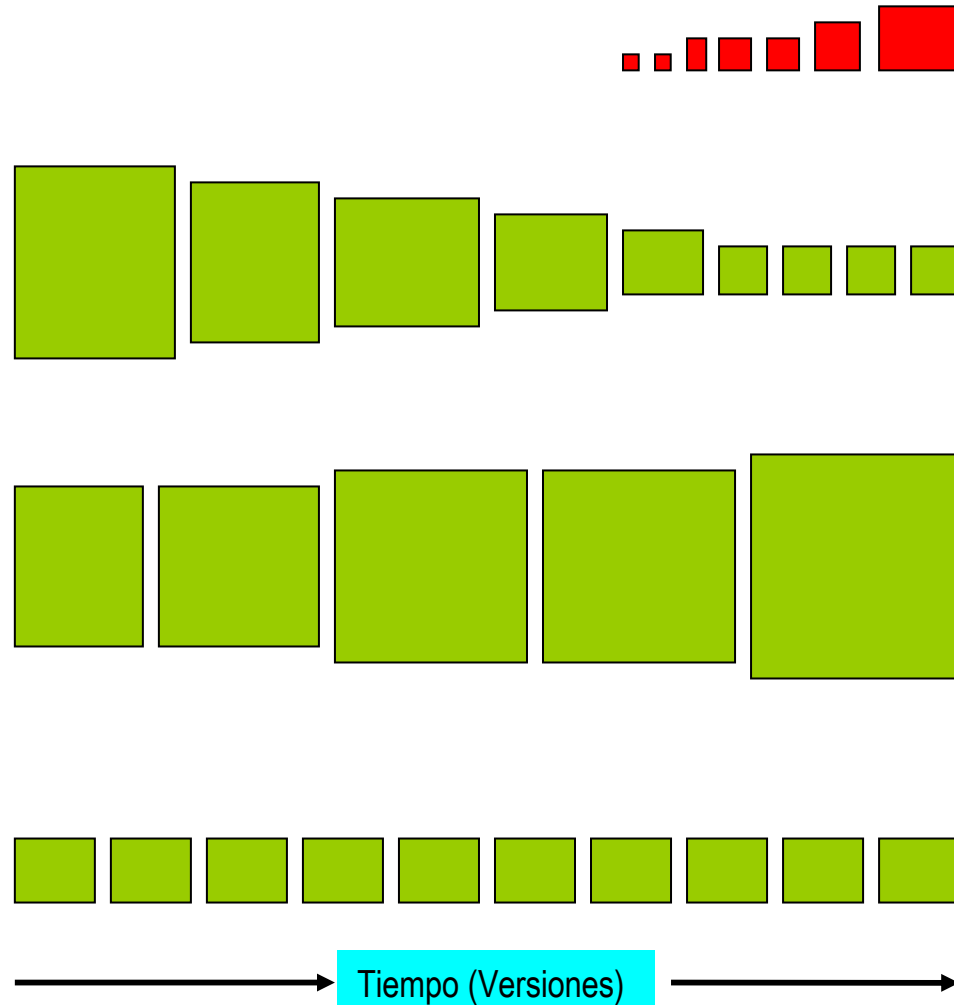
- Se perdió la funcionalidad que tenia y fue perdiendo significado real
- Posiblemente codigo muerto

Gigante Rojo

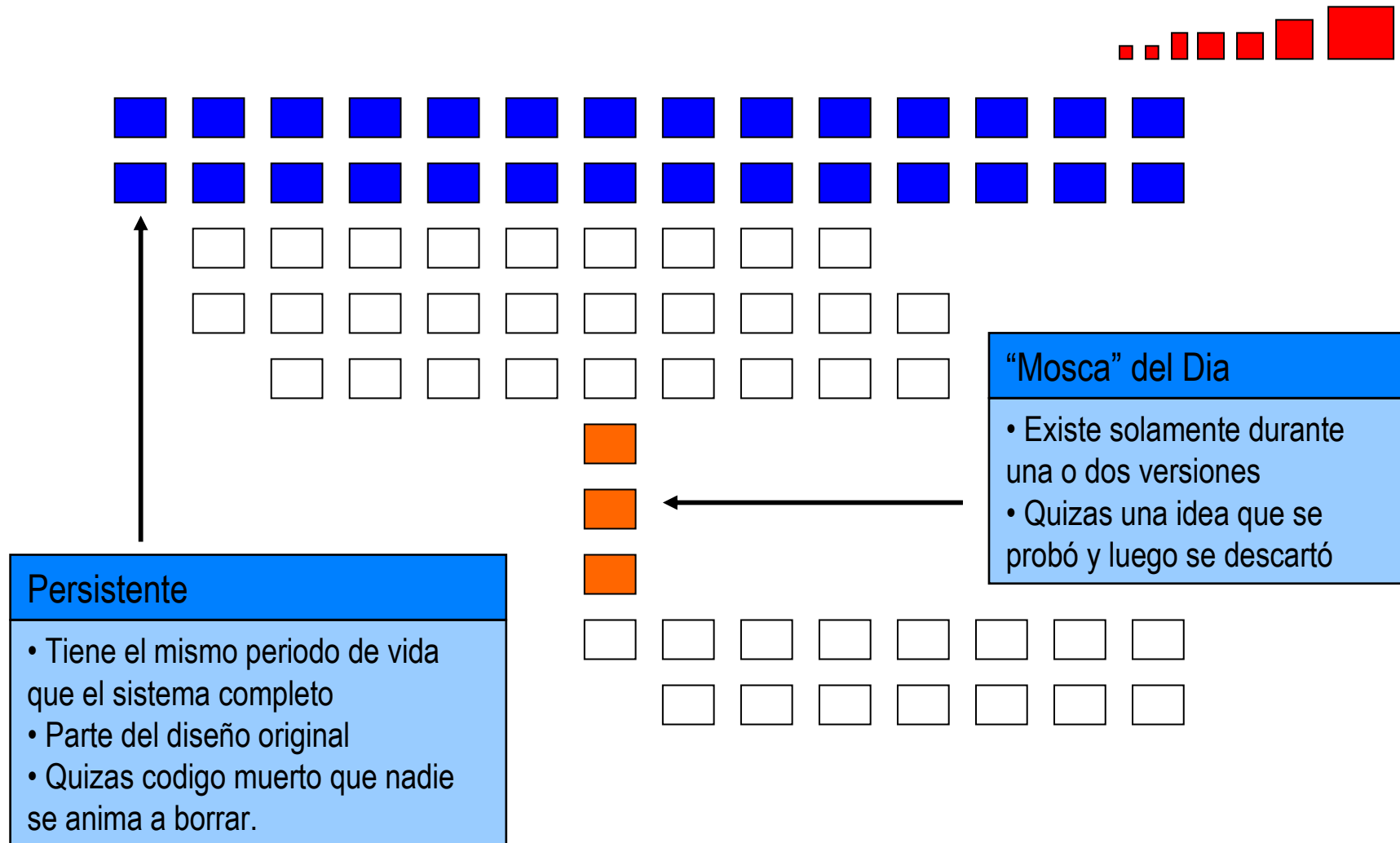
- Un clase “God” que siempre es grande.

Ociosa

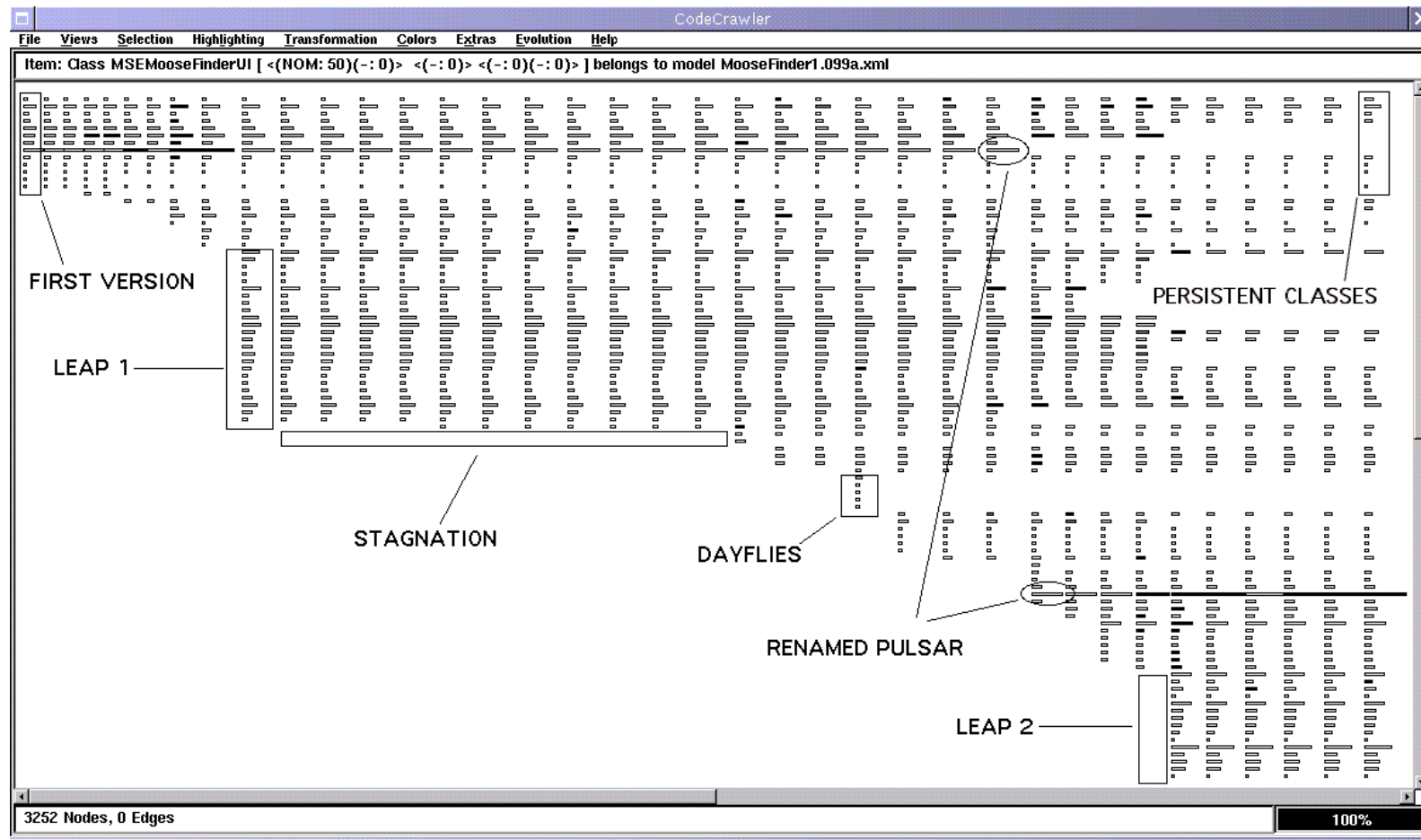
- Mantiene tamaño en diferentes versiones.
- Posiblemente codigo muerto o posiblemente buen codigo.



La Matriz de Evolucion: Lenguaje de Patrones



La Matriz de Evolucion: Lenguaje de Patrones



Conclusiones (Vistas Evolutivas)

➤ Beneficios

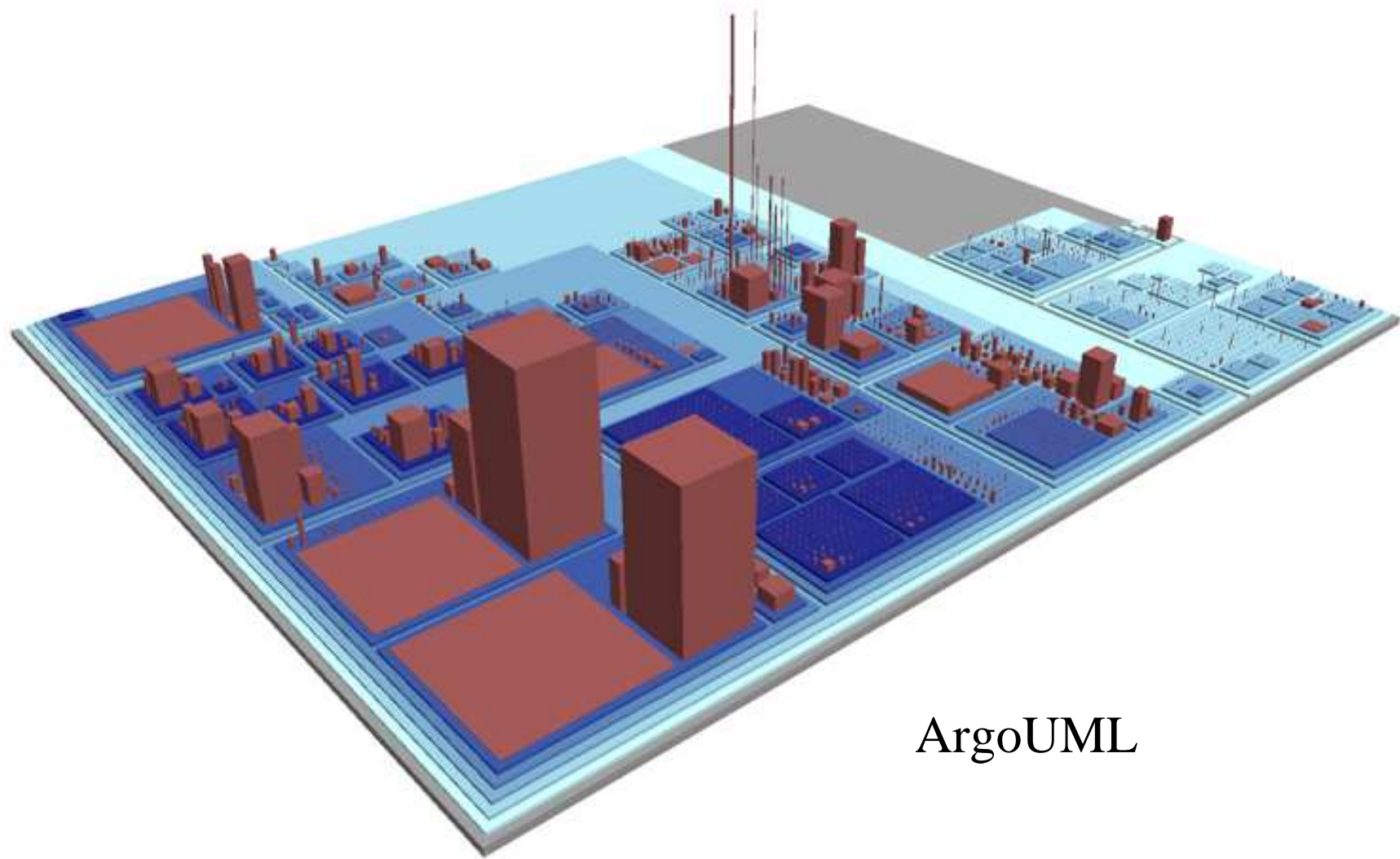
- Reducir complejidad



➤ Limitaciones

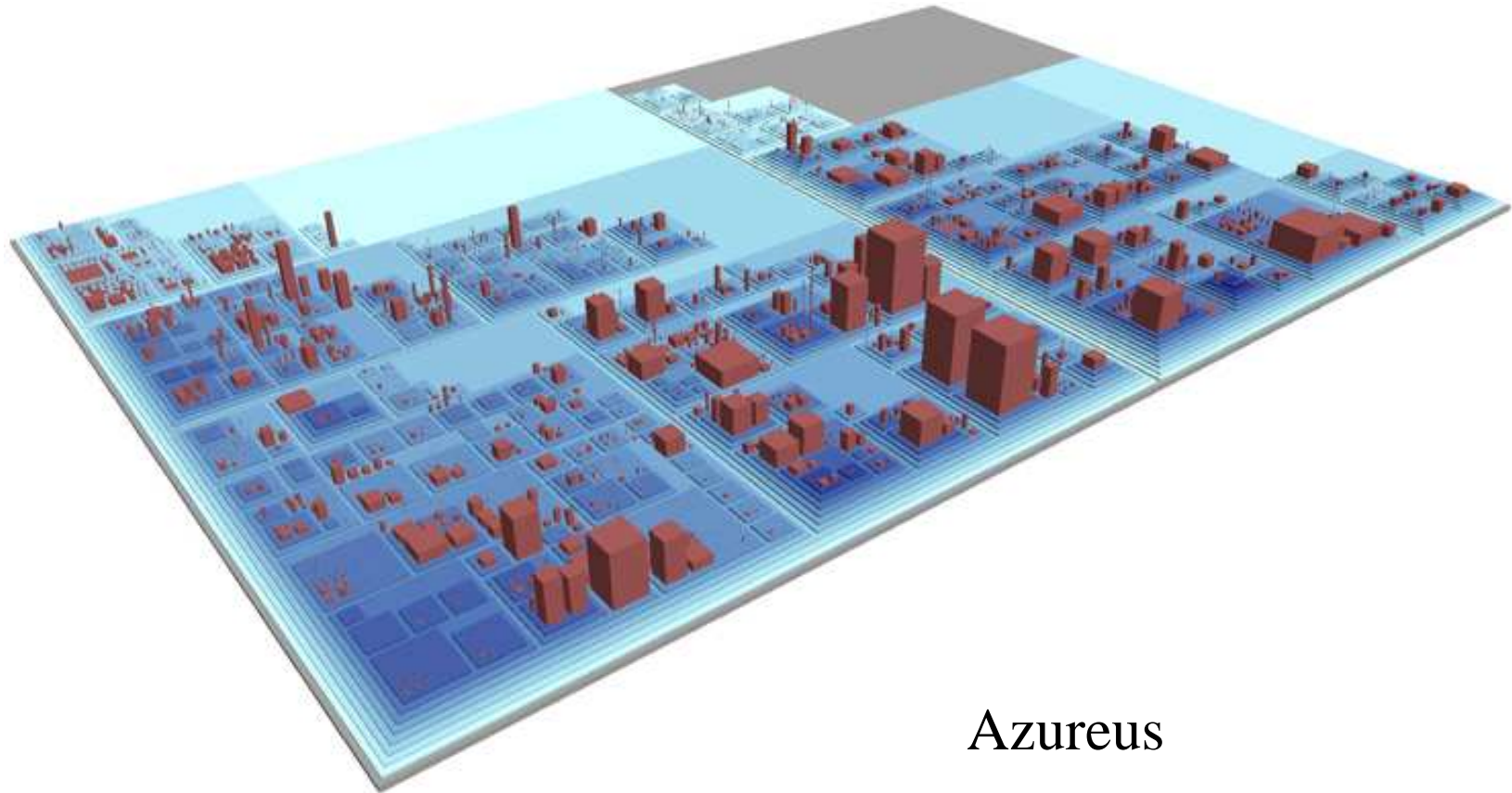
- Escalabilidad
- Problemas de Nombre
- Cambios relativos difíciles de ver

Visualización 3D y Métricas



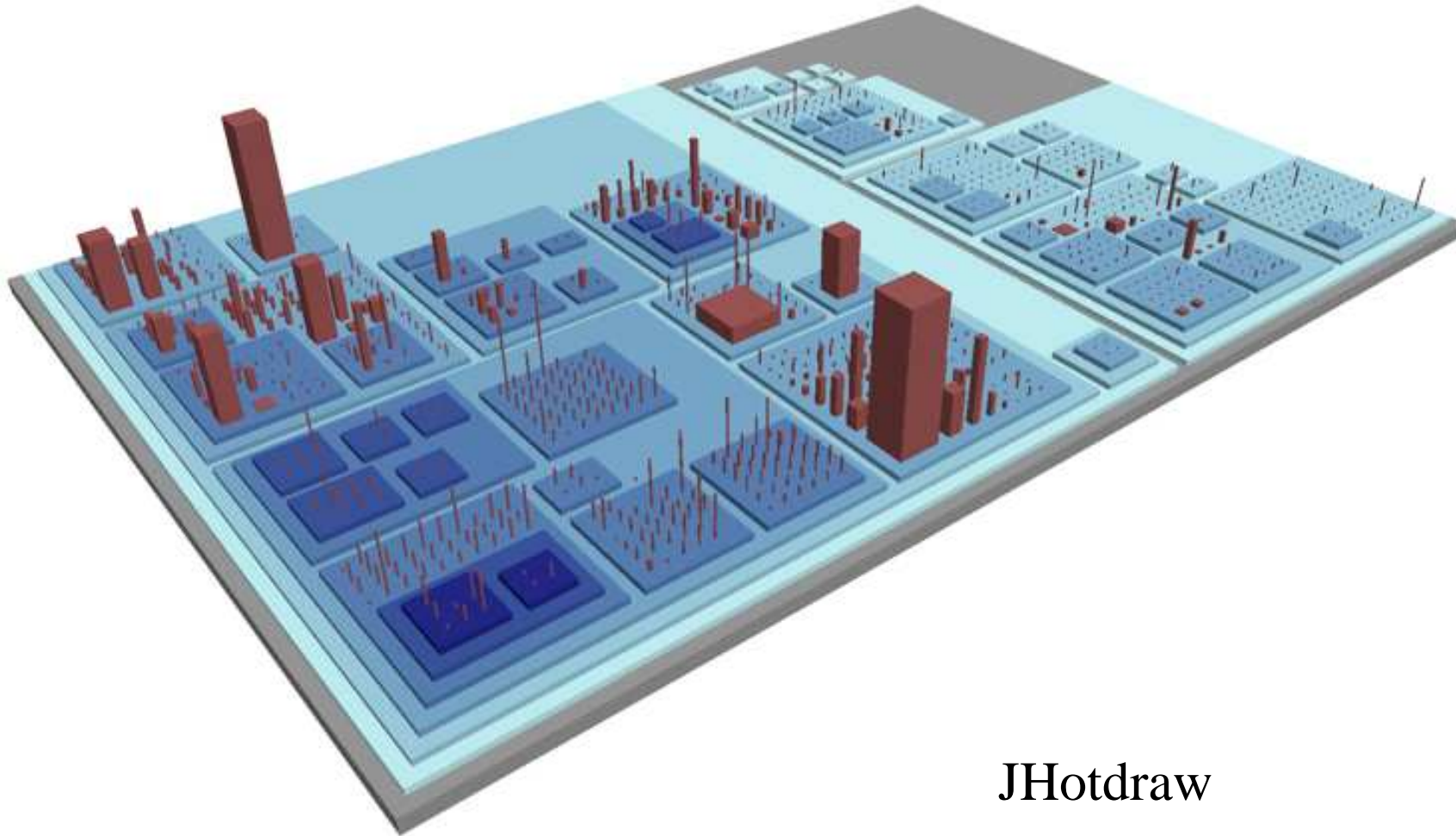
ArgoUML

Visualización 3D y Métricas



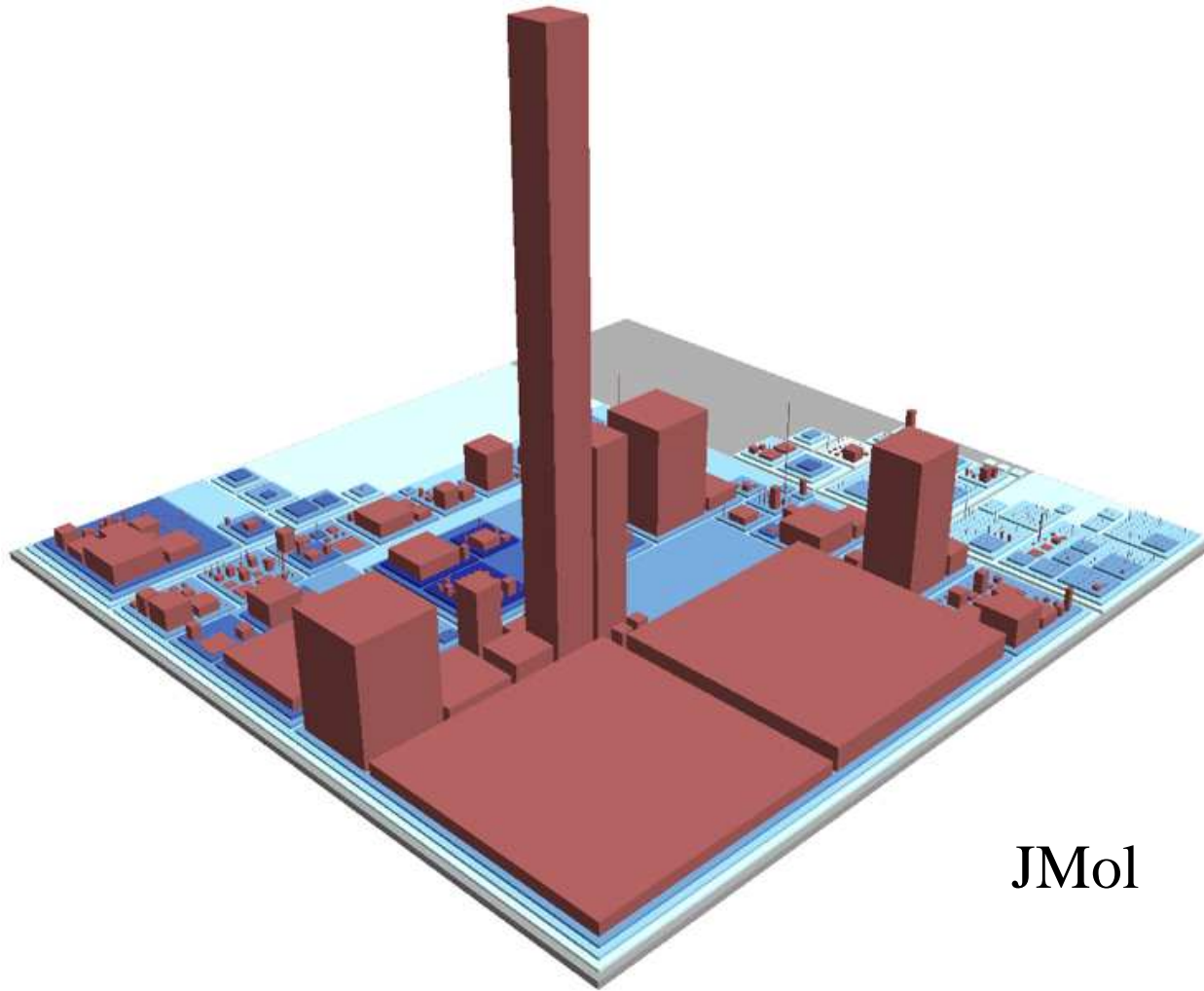
Azureus

Visualización 3D y Métricas



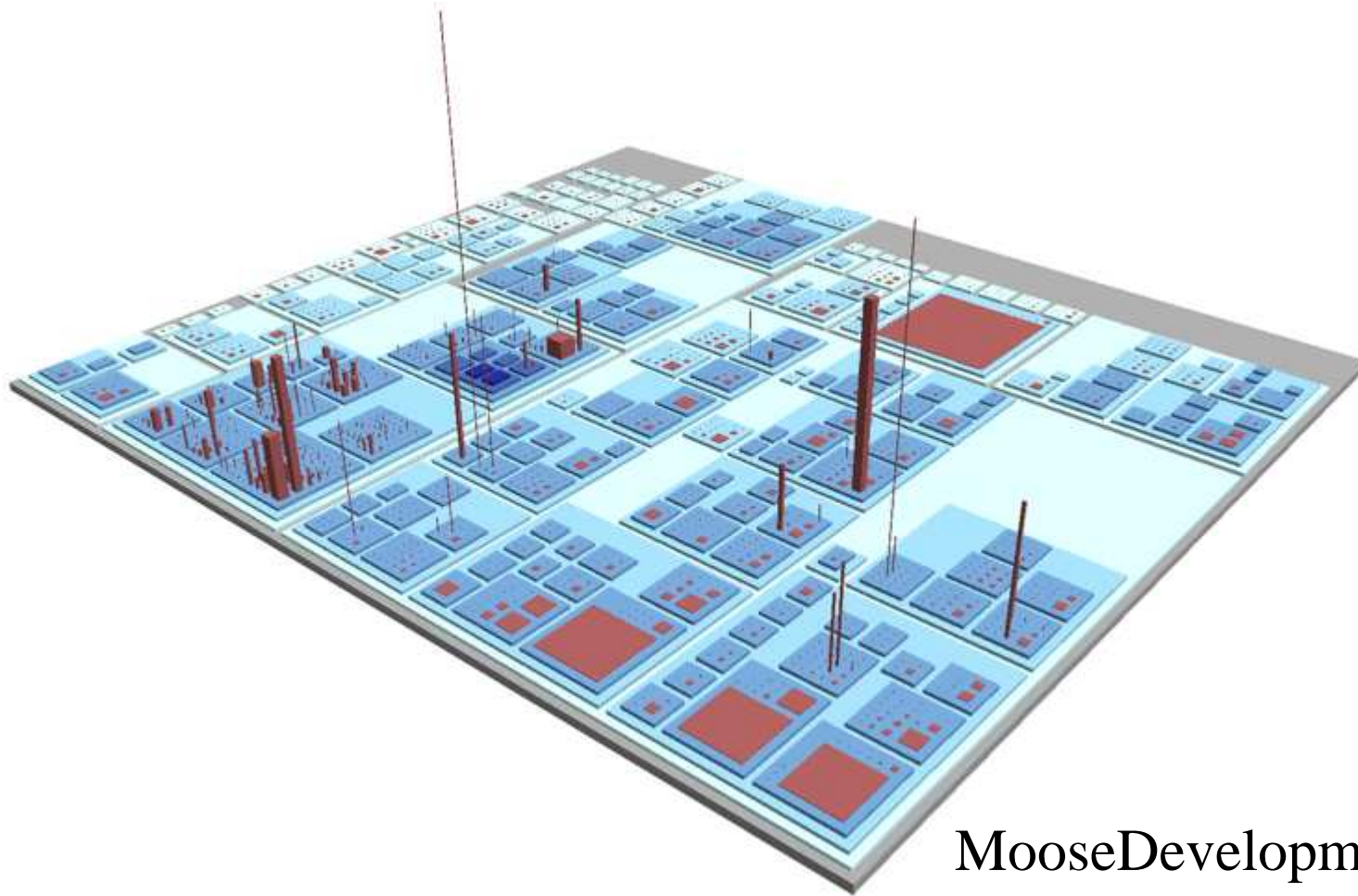
JHotdraw

Visualización 3D y Métricas



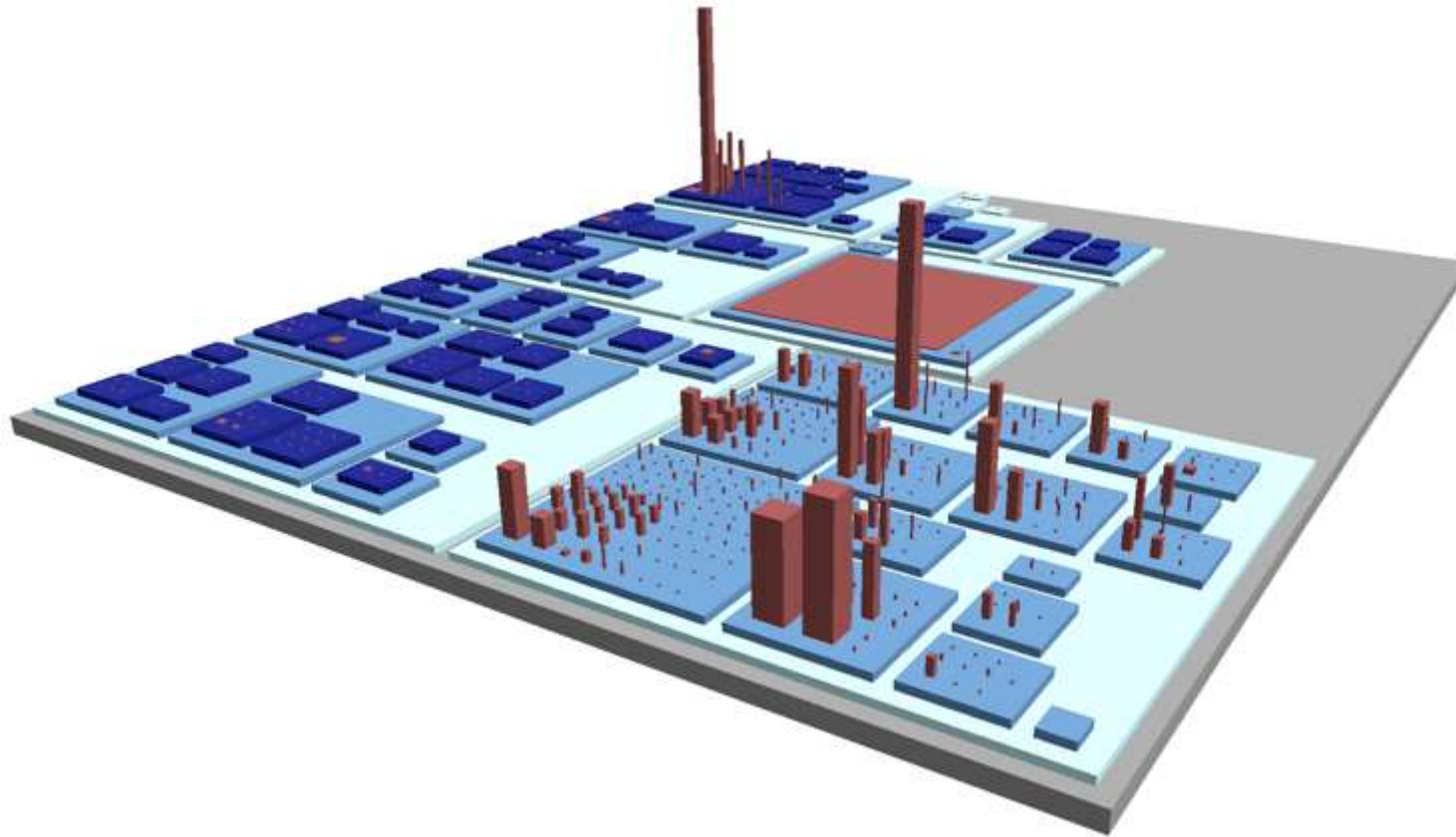
JMol

Visualización 3D y Métricas



MooseDevelopment

Visualización 3D y Métricas



CodeCity