# Objetos, aspectos y algo más: nuevas tendencias en programación

ULISES JUÁREZ MARTÍNEZ, PHD

TECNM / INSTITUTO TECNOLÓGICO DE ORIZABA

# Agenda

- Motivación
- Objetos
  - Objetos funcionales
- Aspectos
  - Relaciones referenciales
- Programación naturalística
  - Tipos naturalísticos
- Lenguajes naturalísticos
  - Pegasus
  - Sicut Naturali (SN)
- Conclusiones

# Motivación

# Ordenación de una lista

```
 1 void bubbleSort(int[] arr) {
 2 int n = arr.length;
 3   int temp = 0;
 4   for(int i = 0; i < n; i++) {
 5     for(int j = 1; j < (n-i); j++) {
 6       if(arr[j-1] > arr[j]) {
 7         temp = arr[j-1];
 8         arr[j-1] = arr[j];
 9         arr[j] = temp;
10 }}}}
```

# Ordenación de una lista

"Repeat the following, until the list is sorted: Go through the list from the beginning till the end. Whenever the actual element is bigger than the following exchange them. If during a pass no exchange occurred, the list is sorted."

# Objetos

EL PROBLEMA SE MODELA CON ENTIDADES DEL MUNDO REAL

# Ventajas de los objetos

## Encapsulación

- Protección de datos

## Herencia

- Reutilización de clases

## Polimorfismo

- Diferente comportamiento con el mismo nombre

# Desventajas de los objetos

## Concurrencia

- Los objetos **no son** concurrentes

## Herencia

- No es posible limitarla o desactivarla

## Composición

- Asociada al acoplamiento
- Preferible la agregación – inyección de dependencias

# Desventajas de los objetos

Requerimientos no funcionales

- No se cuenta con el soporte adecuado para su encapsulación y reutilización
- Se dispersan entre los objetos
- Interfieren con la funcionalidad básica del sistema

# Objetos funcionales

## Programación funcional

- Técnica de programación donde los sistemas se construyen utilizando funciones matemáticas
- Paradigma declarativo – enfoque en el “qué”
- Sistemas robustos con facilidad de prueba y mantenimiento

## Objetos + funciones = objetos funcionales

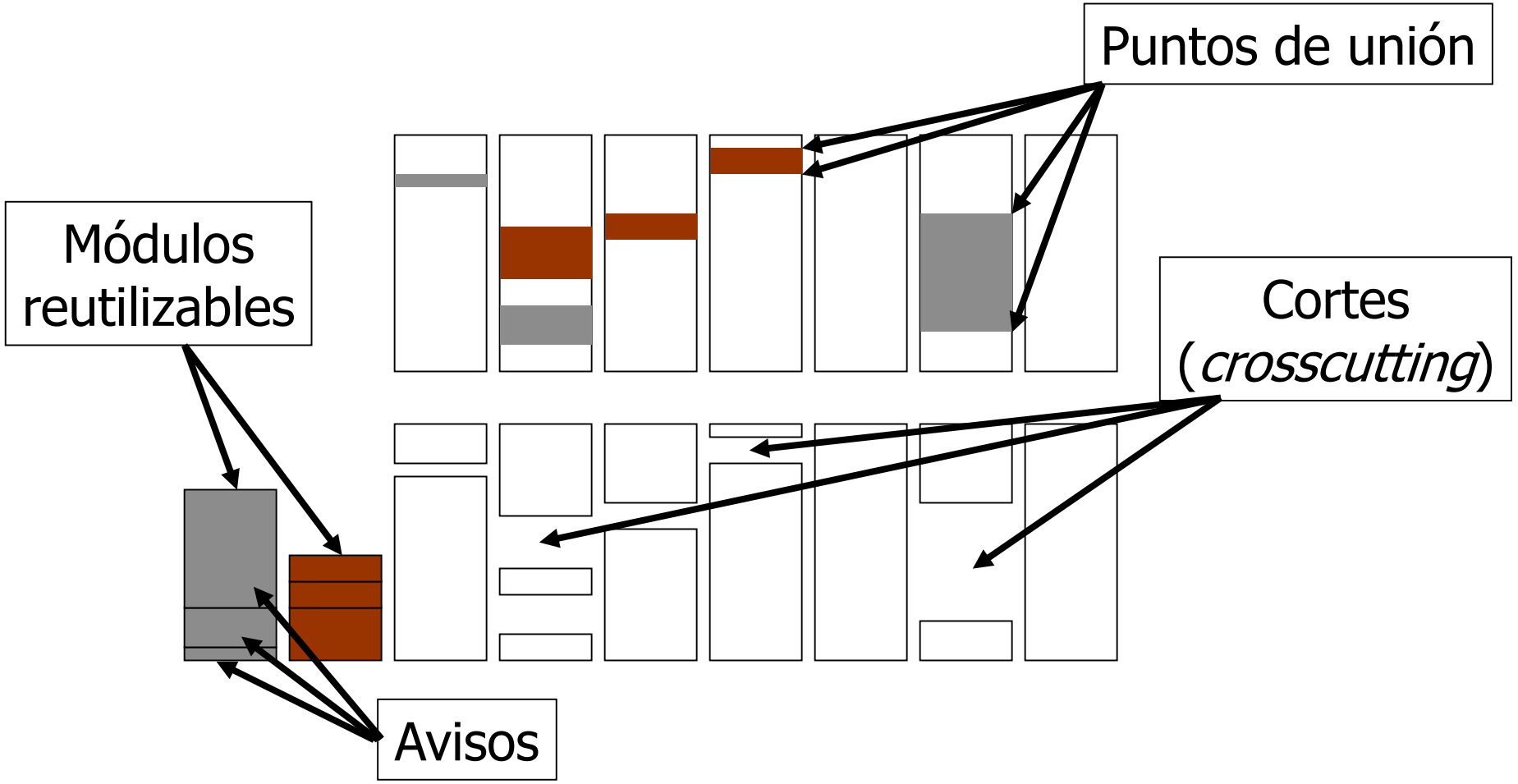
- Scala, Java, Javascript, Python, etc.

## Presentan las mismas ventajas y desventajas

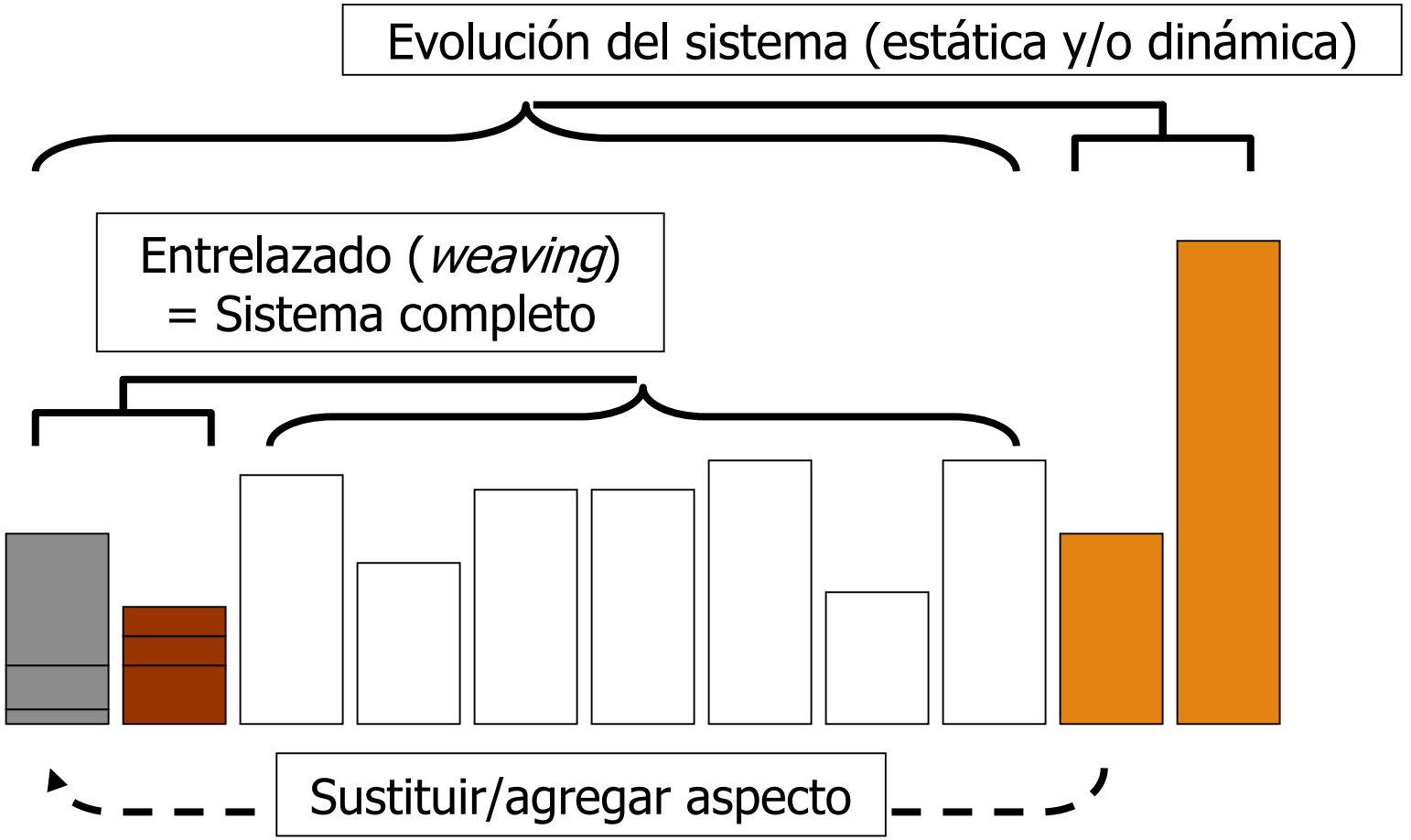
# Aspectos

REUTILIZACIÓN DE REQUERIMIENTOS NO FUNCIONALES

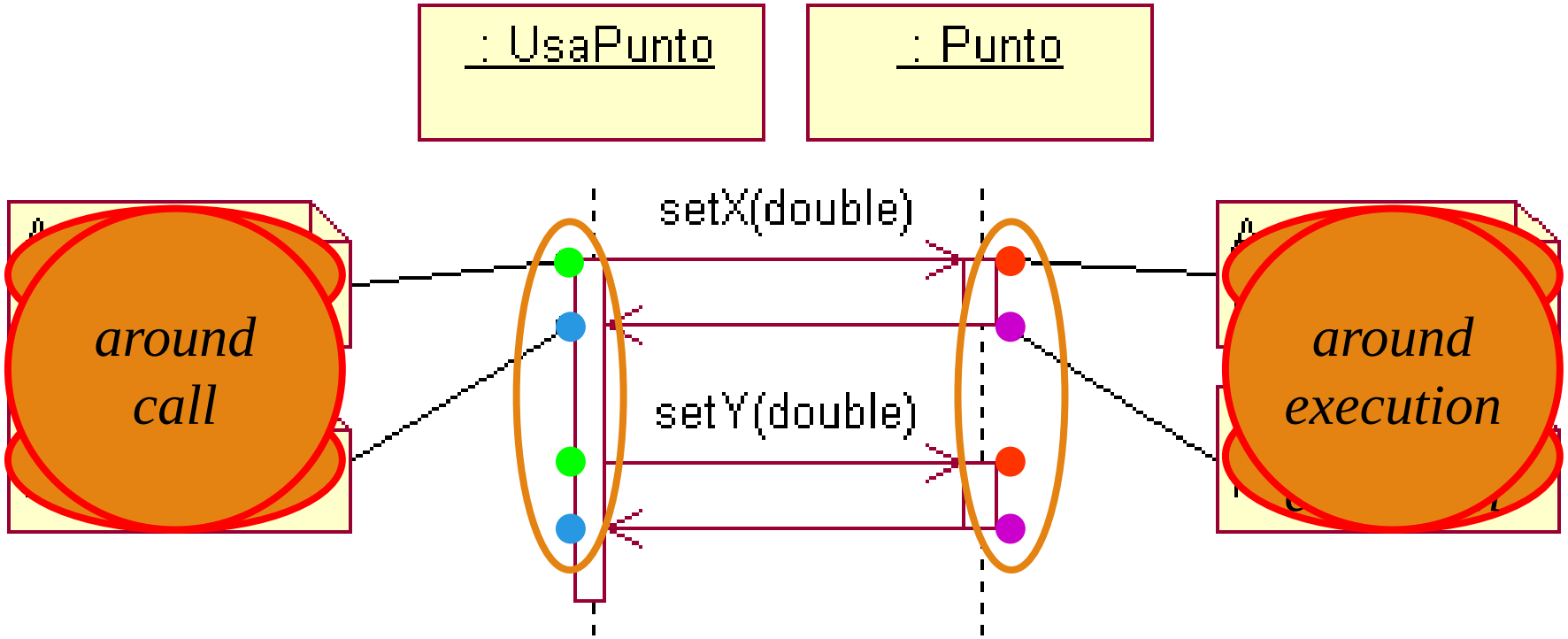
# Aspectos en forma visual



# Evolución de sistemas



# Avisos en el lenguaje AspectJ



# Código de aspectos

```
 1 public aspect Cronometraje {
 2   private long inicio, fin;
 3   pointcut ataque():
 4     execution(void Catapulta.lanzarRocas());
 5   before(): ataque() {
 6     inicio = System.nanoTime();
 7   }
 8   after(): ataque() {
 9     fin = System.nanoTime();
10     System.out.println("Tiempo de ataque: " +
11       fin - inicio) + " nanosegundos");
12 }}
```

# Ventajas de los aspectos

## Reutilización de requerimientos no funcionales

- Incremento en la modularidad de sistemas orientados a objetos
- Facilidad de razonamiento en cada parte del problema

## Complementan el modelo de objetos

- Se conoce como paradigma ortogonal

# Desventajas de los aspectos

No es un paradigma independiente

- Requiere de objetos para generar el comportamiento deseado
- Todo queda representado en clases (bytecode)

Problemas de fragilidad

- El código de aspectos es dependiente del código de objetos
  - Dependencia sintáctica

# (des)Ventajas sintácticas de los aspectos

"Para todos los métodos que lanzan rocas registrar el tiempo de ejecución"

```
3    pointcut ataque():
4      execution(* *.lanzarRocas(..));
5    before(): ataque() {
6      inicio = System.nanoTime();
7    }
8    after(): ataque() {
9      fin = System.nanoTime();
```

Referencia temporal

Referencia estructural

Referencia estructural

"Registrar el tiempo de ejecución cuando se lancen rocas"

# Relaciones referenciales de los aspectos

## Relación estructural

- before
- after
- around

## Relación temporal

- Todos o algunos: `*`, `lanzar*`
- Argumentos: `(..)`, `(*,..)`, `(int, *, ..)`

# Relaciones referenciales de los aspectos

Relación temporal – primitivas de corte en AspectJ

- execution, call
- get, set
- initialization, staticinitialization, preinitialization
- handler
- args, this, target
- within, withincode
- if, cflow, cflowbelow
- adviceexecution

- Esos mecanismos también están cercanos al lenguaje natural (naturalísticos)
- Usar ese tipo de relaciones referenciales al escribir en español, inglés, portugués o hebreo

# Programación naturalística

ESCRIBIR PROGRAMAS CON AYUDA DEL LENGUAJE NATURAL

# Tipos naturalísticos

## Antecedentes

- Objetos, funciones, lógica y aspectos
  - Reflejan una faceta de cómo piensan las personas
  - Permiten comunicar información al igual que el lenguaje natural
- Programación lógica
  - Se basa en la lógica de primer orden
  - Usa la lógica para el planteamiento de problemas y el control sobre las reglas de inferencia para alcanzar la solución automática
  - Lenguaje Prolog

# Tipos naturalísticos

## Definición

- Un tipo naturalístico es un conjunto de cualidades que todas las instancias deben cumplir para pertenecer a ese tipo
  - Una instancia es “de un tipo” o no
  - Conjunto de cualidades – representables por predicados lógicos
  - Instancias – “del” tipo respectivo

# Tipos naturalísticos

Una instancia puede ser de cierto concepto

- “una casa”

Propiedades

- “grande” o “hermoso”

Restricciones adicionales

- “Una casa con una puerta de madera marrón”
- “Una casa que se encuentra junto al río”

# Ejemplos

| NATURALÍSTICO | JAVA |
| --- | --- |
| Jerarquía | |
| ◦ (a house) is (a building) | ◦ class House extends Building {} |
| Propiedades/atributos | |
| ◦ (the house) is red | ◦ assert house.color == Color.RED; |
| ◦ beautiful modern house | |
| ◦ not beautiful modern house | |

# Ejemplos

NATURALÍSTICO

## Cuantificación

- three houses
- two or more houses
- (a house) has (residents and an owner)

## Condiciones

- a beautiful modern house (where **(**some window**)** is open)
- two integers (**(**which**)** are divisible by **(**10**)**)

# Lenguajes naturalísticos

# Pegasus

Es un lenguaje de programación natural

- La codificación es posible utilizando lenguaje natural en alemán, inglés, chino, hindi, español, ruso y otros idiomas

Utiliza:

- Concepto de idea
- Diccionario como elemento de memoria
- Biblioteca para el significado y conocimiento semántico
- Generador de código Java
- Base de datos grande para conocimiento léxico

# Pegasus

## Ejemplos

- “Delete contact data of an employee five years after this employee has left the university.
- “Take the row $\Sigma(1/n^2)$. Print “convergence”, if the row is convergent.”
- “In X, Y, and Z there are coal-fired power stations. In every city there is a transformer station. Connect the power stations with the nearest high voltage power line. Then start them running.”

# Sicut Naturali (SN)

Es un lenguaje de programación naturalístico de propósito general

- Desarrollado en el I. T. Orizaba
  - Tesis doctoral
- Basado en un modelo conceptual
  - Define los elementos mínimos para un paradigma naturalístico
  - Permite el desarrollo de lenguajes naturalísticos de propósito general
- El compilador genera bytecode para la JVM
  - Utiliza Scala y AspectJ

# Sicut Naturali (SN)

## Modelo conceptual

- Elementos mínimos
  - Substantivo (singular y plural), adjetivo, verbo y circunstancia (eventos)
  - Sintagma (oraciones complejas)
  - Anáfora (referencia a elementos descritos previamente)
- Elementos opcionales
  - Deixis completa (referencia a elementos descritos antes y después del texto)
  - Indicadores (abstracciones)
  - Tipificación basada en propiedades (agregar nuevas propiedades)

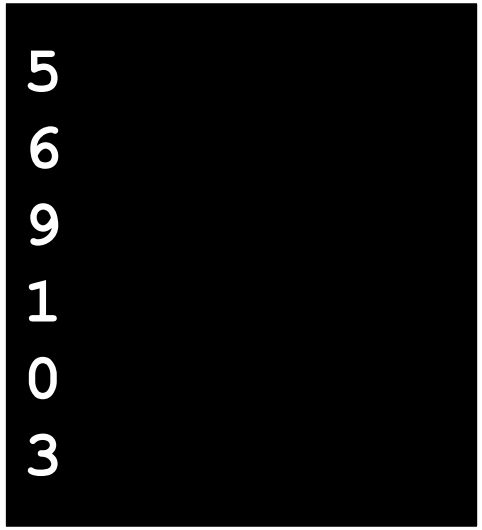
# Sicut Naturali (SN) – Ejemplos

```
main Sumatoria:
    an Integer Number with 10 as value.
    an Integer Number with 25 as value.
    an Integer Number with 34 as value.
    add the second Number to the first Number.
    System prints it and newline.
    add the first Number to the third Number.
    System prints it.
```

```
35
69
```

# Sicut Naturali (SN) – Ejemplos

```
main Plurales:
    number are Numbers.
    plural add 5 to numbers.
    plural add 6 to numbers.
    plural add 9 to numbers.
    plural add 1 to numbers.
    plural add 0 to numbers.
    plural add 3 to numbers.
    repeat the next instruction for each
      element of numbers as number.
    System prints number and end.
```



# Conclusiones

Los objetos tienen limitaciones

La programación orientada a aspectos ayuda y complementa a los objetos

La sintaxis derivada de programas de aspectos son la base para expresar ideas en forma cercana a un idioma

## Programación naturalística

- Nuevo enfoque que permite programar utilizando formas controladas de un lenguaje natural
- Pegasus es un lenguaje naturalístico que utiliza diversas tecnologías para lograr programar de forma natural
- SN es un lenguaje naturalístico de propósito general basado en un modelo naturalístico que no requiere de tecnologías adicionales
- Fuerte interés en especificación de requerimientos
  - La especificación es el programa a ejecutar

# ¿Preguntas?

ujuarezm@orizaba.tecnm.mx