Paradigmas de programacion

José Galaviz

Programar

- Crear un programa que pueda ser ejecutado por un dispositivo de cómputo.
- Es un medio para resolver problemas.
  - Si la solución puede encontrarse o aproximarse por medio de un algoritmo.

Un programa

Es la descripción precisa (sin ambigüedad en términos del *lenguaje inherente* del dispositivo de cómputo) del proceso que se desea ejecutar.

Lenguajes

- Todo dispositivo de cómputo posee un lenguaje inherente (su lenguaje de máquina).
- Cualquier dispositivo de cómputo es, a lo más, una máquina de Turing.
- Así que todos los lenguajes (Turing completos) son equivalentes.

Máquinas virtuales

- Cuando programamos en C++ estamos suponiendo una máquina virtual que entiende C++ (aunque en los hechos lo traducimos al lenguaje de la máquina real).

Descripción del proceso (el programa)

- Para describir el proceso que necesitamos implementar debemos ubicarnos primero en una de estas máquinas virtuales.
- Que no es otra cosa que una abstracción.

Abstracción

- Como en toda abstracción hay conceptos y elementos esenciales: *un marco de referencia*.
- Como toda abstracción, pretende fragmentar la realidad para hacer manejable la complejidad.
- El marco conceptual determina con base en qué, se hará el modelo simplificado del proceso.

Paradigma

- Es justamente el marco conceptual en el que inscribimos la construcción del programa.
- Nos dice en qué debemos enfocarnos para simplificar el programa.

Origen de “paradigma”

- Robert Floyd, Stanford University, 1979.
- Patrón, ejemplo, arquetipo. Una idea de acuerdo a la cual se hacen las cosas.

Una primera clasificación

Existen dos tipos muy generales de paradigmas:

- Declarativo.
- Imperativo.

Imperativo

- Partiendo de:
  - Descripción de los datos de entrada, sus valores y sus propiedades.
  - Descripción de los datos de salida, sus valores y sus propiedades.
  - Se describe **cómo** se obtienen los valores de los datos de salida con base en los de entrada.

Declarativo

- Se debe especificar **qué** se debe hacer.
  - Describir los datos de entrada, sus valores y sus propiedades.
  - Describir los datos de salida, sus valores y sus propiedades.
  - Describir la relación entre ambos.
    - Usando ecuaciones o predicados.

Paradigma añejo

- El primer paradigma usado para programar fue el imperativo.
- El lenguaje de máquina y el ensamblador son imperativos.
- También lo son Fortran, Algol, Cobol, etcétera.

Mejoras

- Es mejor agrupar segmentos de código que puedan ser invocados varias veces para reutilizarlos.
- Usar estructuras de control que indiquen qué se hace sin tener que seguir el código.
- **Programación estructurada**.
- 1968, E. Dijkstra, *GoTo Statement considered harmful.*

Paradigma procedural

- Se puede robustecer el esquema definiendo procedimientos, rutinas con protocolos bien establecidos de paso de parámetros y regreso de resultados.
- **Programación procedural**.
- Lenguajes: Fortran, Cobol, Algol, Basic, C (3GL).
- El código puede ser leído por un ser humano.

Programación Orientada a Objetos

- Modelar los objetos presentes en dominio de problema, abstrayendo las características y el comportamiento relevante para resolver el problema.
- **Programación orientada a objetos**.

El paradigma declarativo

- Si el programa pretende establecer cuál es la lógica del cómputo que se pretende realizar, pero no la secuencia de pasos que lo realizan.
- Lenguajes: Prolog, SQL, Lisp.

Sub clasificación del paradigma declarativo

- Programación lógica. Se pretende modelar el dominio del problema usando un sistema formal de la lógica de primer orden: Prolog.
- Programación funcional. Se pretende modelar las funciones que realizan las transformaciones de los datos hasta producir la salida: Lisp, scheme, Haskell.

scheme

```
(define (fib-rec n)
  (if (< n 2)
      n
      (+ (fib-rec (- n 1))
         (fib-rec (- n 2)))))
```

prolog

```
fib(1, 1) :- !.
fib(0, 0) :- !.
fib(N, Value) :-
 A is N - 1, fib(A, A1),
 B is N - 2, fib(B, B1),
 Value is A1 + B1.
```

Java

```java
public static long itFibN(int n) {
    if (n < 2)
     return n;
    long ans = 0;
    long n1 = 0;
    long n2 = 1;
    for(n--; n > 0; n--) {
     ans = n1 + n2;
     n1 = n2;
     n2 = ans;
    }
    return ans;
}
```

Prolog

```
ama(vicente,mia).
ama(sebastian,mia).
ama(victoria,ricardo).
ama(ricardo,victoria).

celoso(X,Y):-  ama(X,Z),  ama(Y,Z).
```

```
?-  celoso(sebastian,W).
W  =  vicente
```

scheme

```
(define (my-sqr x) (* x x))


(define (fact n)
  (if (= n 0)
      1
      (* n (fact (- n 1)))))
```

Diferencia de modelos en la imperativa

- La programación estructurada o procedural modela el algoritmo, las acciones del proceso.
- La programación orientada a objetos modela las entidades involucradas en el problema a resolver.

Sub clasificación del paradigma imperativo

Para lidiar con la complejidad de lo que se modela...

- La programación estructurada o procedural pretende dividir el proceso en algoritmos cada vez más y más pequeños y simples.
- La programación orientada a objetos pretende dividir el proceso de acuerdo con las distintas entidades involucradas.

Evolución

- Durante las últimas décadas del siglo XX cambiaron significativamente el uso y los usuarios de computadoras.
- Ahora se requería software más amigable, para hacer tareas antes impensables: multimedia, sistemas de ventanas, ambientes gráficos.
- Ya no sólo usuarios especializados.

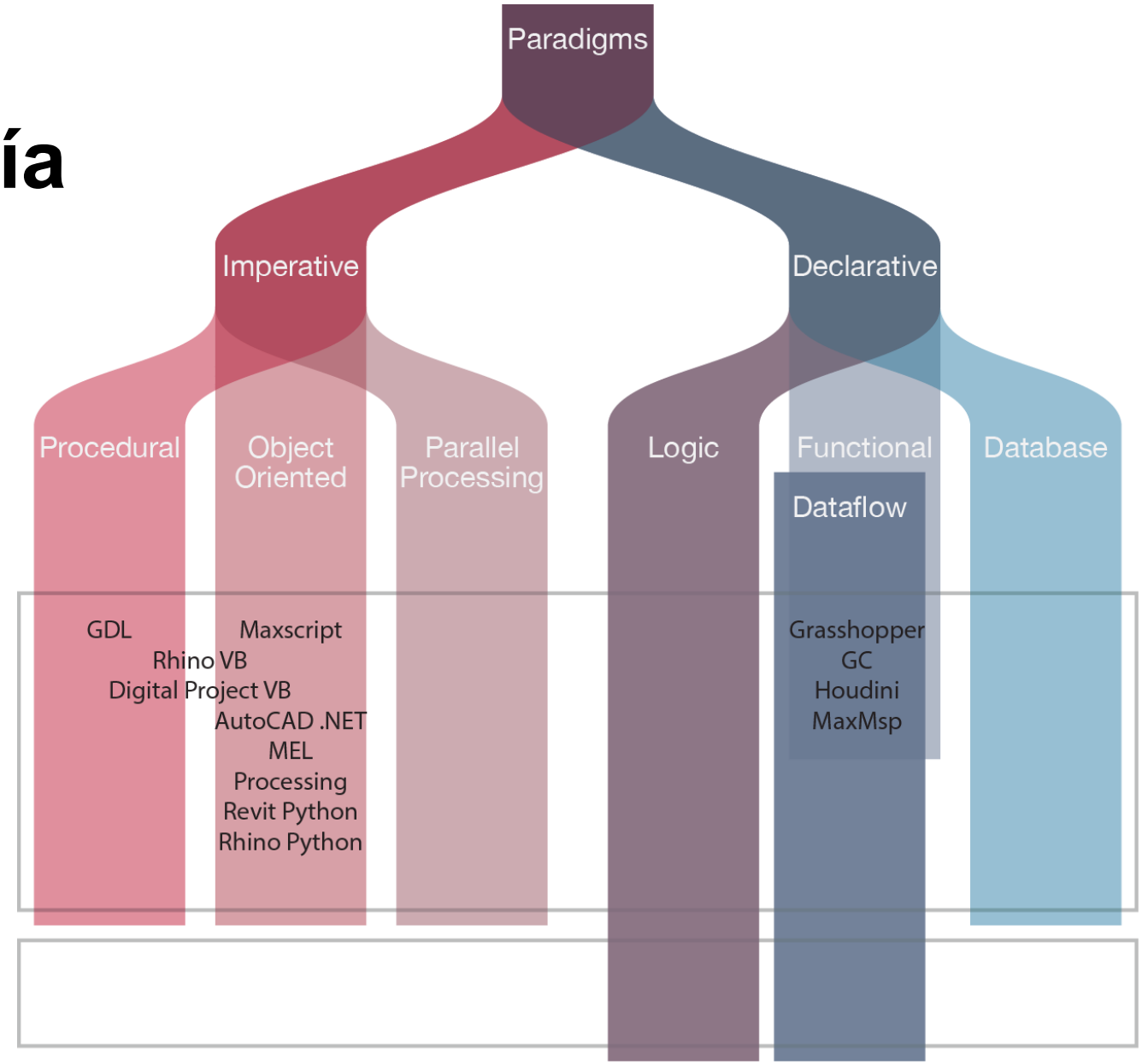
Lo bonito cuesta

- Las nuevas demandas del software hicieron que la complejidad de este creciera.
- La descomposición en términos algorítmicos dejó de ser opción.
  - Muchas tareas, muy complejas, con muchos efectos colaterales e interacciones y dependencias entre ellas.

De allí la Programación Orientada a Objetos

- Resulta más sencillo buscar cuales son las entidades (objetos) que intervienen en el problema.
- Abstraer sus atributos relevantes.
- Su comportamiento.
- O sea Tipos de Datos Abstractos.
- Cada objeto debe asumir las responsabilidades que le tocan según su clase.

Taxonomía


Paradigms
Imperative
Declarative
Procedural
Object Oriented
Parallel Processing
Logic
Functional
Database
Dataflow
GDL
Rhino VB
Digital Project VB
Maxscript
AutoCAD .NET
MEL
Processing
Revit Python
Rhino Python
Grasshopper
GC
Houdini
MaxMsp

Programación OO: nomenclatura

- Objeto: Entidad en el dominio del programa que existe sólo en tiempo de ejecución y posee
  - Datos. En forma de sus atributos o variables de estado (o campos en terminología vieja).
  - Comportamiento. Establecido por los métodos que posee, los servicios que implementa.
  - Identidad. Una manera de referirse a él en tiempo de ejecución.
- Son *instancias* o ejemplares de una clase.

Encapsulación

- Ocultar la complejidad.
- Los detalles innecesarios al exterior.
- El cómo, dejando sólo el qué.
- Para que nadie dependa del cómo.
- Y entonces el programador pueda cambiarlo sin echar a perder a los que usan sus clases.

Reusabilidad

- Que el software previamente hecho pueda usarse para nuevas construcciones.
  - Objetos que contienen otros objetos: composición.
  - Objetos que son especialización de otros: herencia.
  - Objetos que usan los servicios de otros para cumplir con los suyos.