Paradigmas de Programacion

FIUBA - Técnicas de Diseño

Un poco de historia...

- 1930 - Problema de decisión de David Hilbert
  ¿Existe un **modelo matemático** de **computación**?

- 1933 - Kurt Gödel define las **Funciones Recursivas Generales**.

- 1936:
  - Alan Turing define la **Maquina de Turing**
  - Alonzo Church define el **Calculo Lambda**

Computabilidad

Tésis de Church-Turing:

- Los tres formalismos tienen la misma capacidad de resolver cualquier cómputo
  (Demostrado mediante simuladores de un formalismo, escrito en otro)
- Todo algoritmo puede expresarse en estos formalismos
  (No demostrado, pero no hay contraejemplos)

Computabilidad

Si cualquier otro formalismo puede simular una Máquina de Turing, decimos que es **Turing-completo**

- ➔ Una computadora es una Máquina de Turing con memoria limitada
- ➔ Un lenguaje de programación suele especificarse como una implementación de un formalismo

Computabilidad

Si todos los lenguajes pueden computar lo mismo,

¿por qué no estandarizamos y usamos uno solo?

Por ejemplo, assembly

Computabilidad

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand"

Martin Fowler

Conceptos

Concepto: Alguna idea que se puede **expresar o prohibir** en un lenguaje

Modelo de Programación: Conjunto de **técnicas de programación** y **principios de diseño** aplicados a un lenguaje

Dado un problema, queremos un lenguaje con conceptos y modelo de programación que faciliten resolverlo

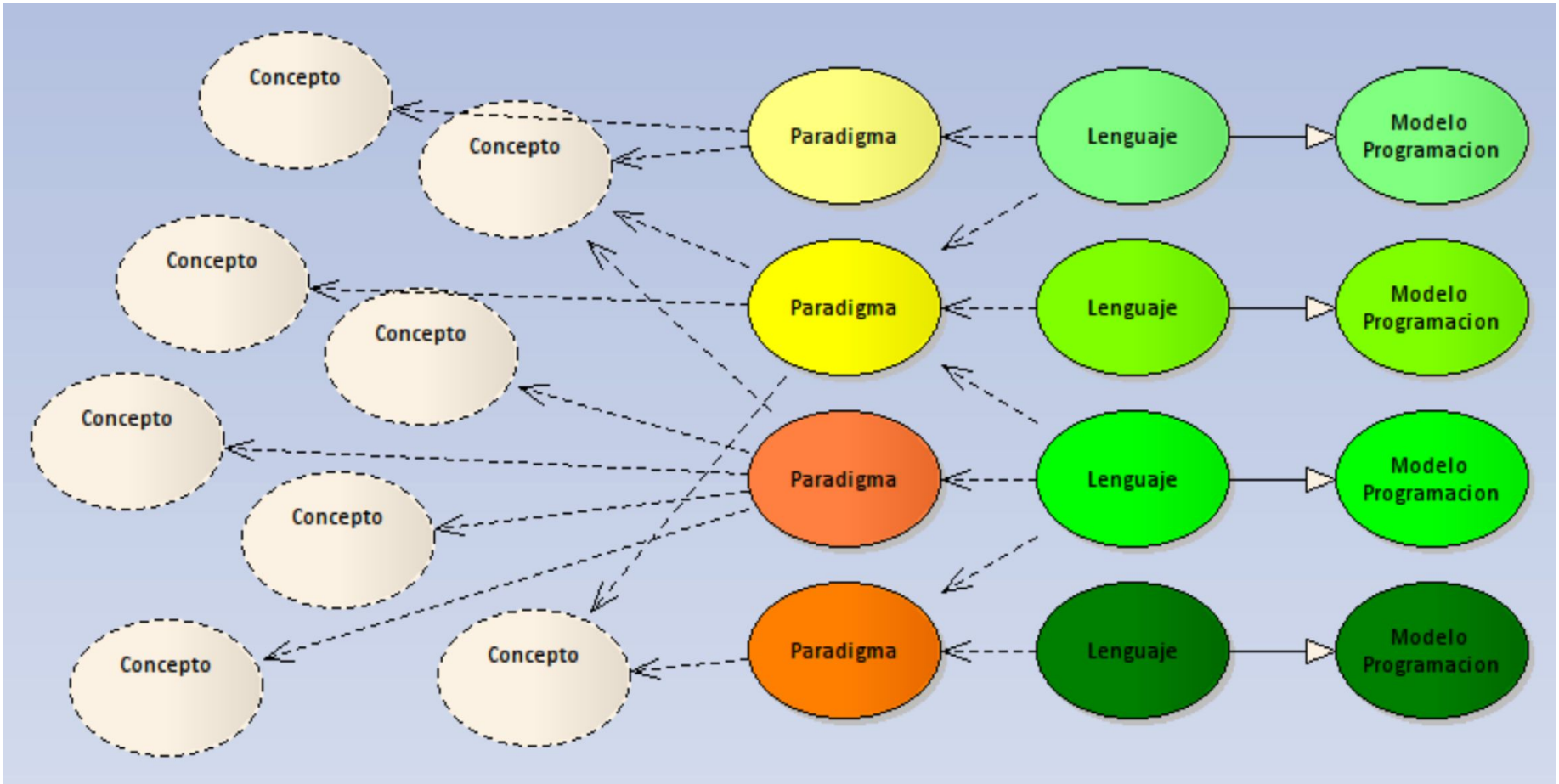
Paradigmas

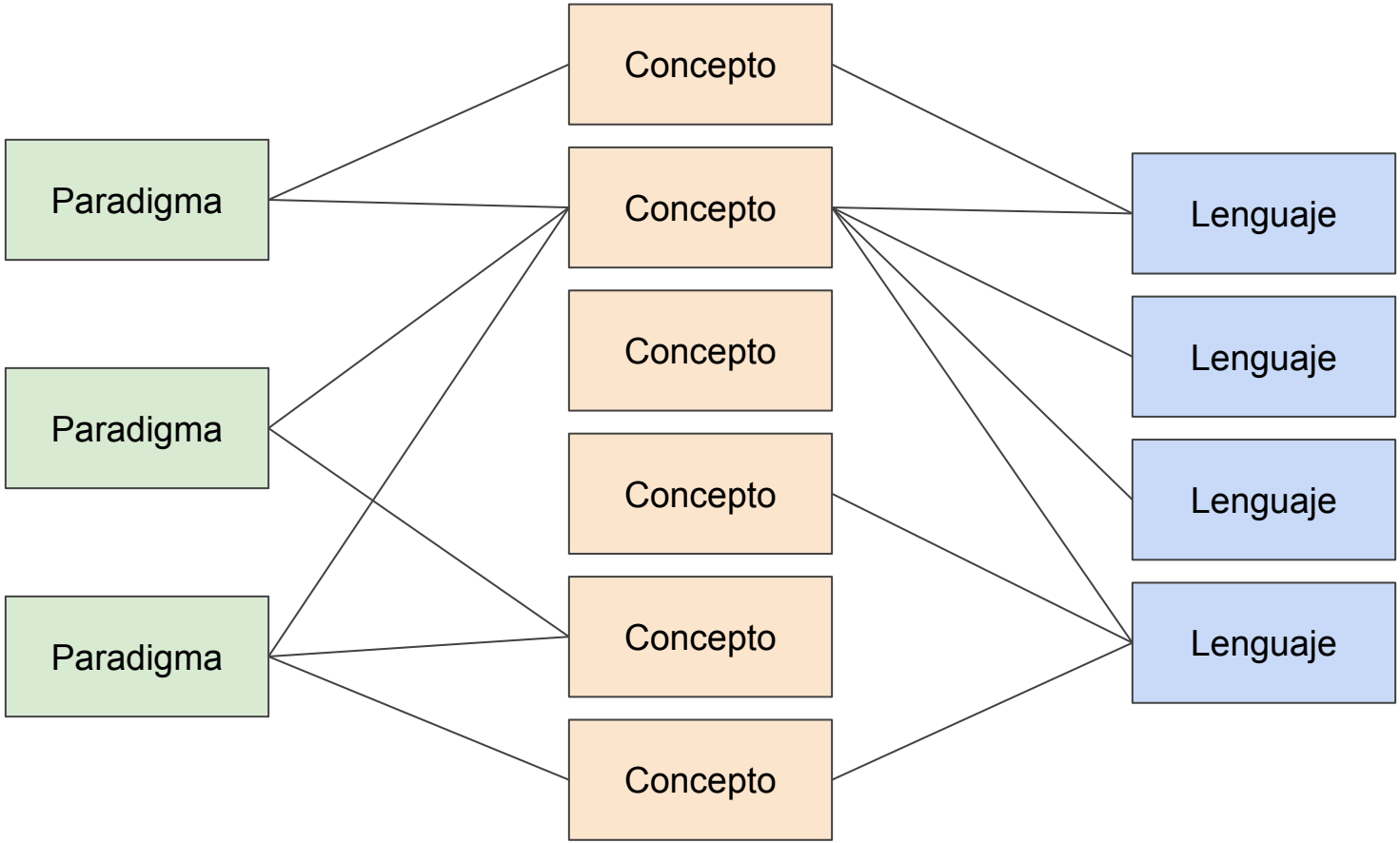
2. m. Teoría o conjunto de teorías cuyo núcleo central se acepta sin cuestionar y que suministra la base y modelo para resolver problemas y avanzar en el conocimiento. *El paradigma newtoniano*.

https://dle.rae.es/paradigma

Paradigma de Programación

- Conceptos centrales
- Mecanismos de razonamiento
  - Conceptos favorecidos
  - Conceptos prohibidos
- Mecanismos de comunicación
  - Vocabulario
  - Notacion


Concepto
Concepto
Concepto
Concepto
Concepto
Concepto
Concepto
Concepto
Paradigma
Paradigma
Paradigma
Paradigma
Lenguaje
Lenguaje
Lenguaje
Lenguaje
Modelo Programacion
Modelo Programacion
Modelo Programacion
Modelo Programacion


Paradigma
Paradigma
Paradigma
Concepto
Concepto
Concepto
Concepto
Concepto
Concepto
Lenguaje
Lenguaje
Lenguaje
Lenguaje

Combinaciones de Paradigmas

Poder tomar múltiplas perspectivas

- ➔ Es bueno para las personas: Múltiples puntos de vista para hallar soluciones
- ➔ Es bueno para componentizar sistemas: Cada uno se expresa en su paradigma natural

- ➔ Es terrible si mezclamos sin criterio: Hay conceptos que interactúan positivamente, y conceptos que interactúan negativamente

- ➔ Más detalles después de hablar de paradigmas concretos!

Un Problema

"The determined Real Programmer can write FORTRAN programs in any language."

Ed Post, Real Programmers Don't Use Pascal, 1982

(En este caso, escribir FORTRAN no era considerado algo bueno)

Imperativo vs Declarativo

Imperativo vs Declarativo

Imperativo

- Describe instrucciones a ejecutarse paso a paso para variar el estado del programa
- El estado final debería ser la solución al problema

Declarativo

- Describe el problema que se quiere solucionar
- El sistema usa esta descripción para intentar hallar un algoritmo que resuelva el problema

Imperativo vs Declarativo

- No es binario, es una escala ordenada
  - "X es más declarativo que Y"
  - "X es declarativo" ¿relativo a qué?

- Todo sistema declarativo requiere una base más imperativa para dar instrucciones al hardware
  - **El usuario puede no tener acceso a estas capacidades**

Otra forma de interpretar Imperativo y Declarativo

Semántica: Significado de un programa

- Semántica Operativa: Estado y transformaciones del mismo
  - Problema: Programador como interprete humano
- Semántica Denotacional: A que objeto matematico corresponde
  - Problema: Suele ser un objeto matemático extremadamente complejo
- Semántica Axiomática: Pre y post condiciones

Otra forma de interpretar Imperativo y Declarativo

```
listaOrdenada = sort(lista);
```

- Semántica Operativa
  - Mismo resultado que si se ejecuta este código: (Código de algún ordenamiento).
  - Existen muchas semánticas operativas posibles, y demostrar que dos de ellas son equivalentes es difícil
- Semántica Axiomática:
  - Pre: `lista` es una lista
  - Post: `listaOrdenada` es permutación de lista
  - Post: Si i, j son indices validos, `i ≤ j ⇔ listaOrdenada[i] ≤ listaOrdenada[j]`

Algunos Paradigmas

Programación Estructurada

- El control de flujo de un programa se puede expresar con tres primitivas
  - Secuencia
  - Alternativa
  - Repeticion
- Cualquier otro control de flujo está prohibido
  - GOTO considered harmful (Dijkstra 1968)
  - Permite determinar localmente los posibles predecesores a la instrucción actual
- Originalmente requería convencer a los programadores a seguir sus buenas prácticas, hoy día se integra por defecto a nivel lenguaje

Programación Modular

Módulo: Conjunto de código con una interfaz definida que cumple un propósito discreto

- Archivos separados de interfaz/implementación, visibilidad, namespaces, etc
- Permite limitar acceso a estado/operaciones
- En general "X es objeto" implica "X es módulo", pero no al revés

Programación Orientada a Objetos

Objeto

- Estado y comportamiento agrupado en una entidad
- Objetos encapsulan su estado, ocultándolo a otros
- Los objetos solicitan a colaboradores que apliquen su comportamiento relevante

- Permite analizar estado/comportamiento de un objeto sin considerar otros

Programación Orientada a Objetos

Estado y comportamiento de objetos:

- Prototipos: Un objeto es replicado para crear un nuevo objeto. Los objetos se modifican para describir el estado/comportamiento deseado
- Clases: Un molde se usa para crear objetos nuevos. Los objetos pueden o no ser modificables (No modificable es más fácil de implementar)

Programación Orientada a Objetos

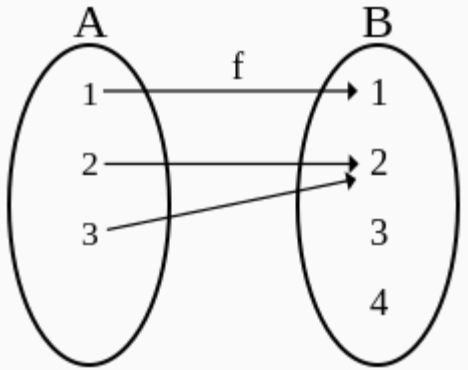
Comunicación entre objetos:

- Mensajes
  - Objeto decide cuándo procesar y cómo responder
  - Típico en sistemas concurrentes/distribuidos (Modelo de actores)

- Métodos
  - Invocador espera la respuesta sincrónicamente
  - Típico dentro de un hilo

Programación Funcional

Basado en **funciones matemáticas**



Al evaluar una función matemática:

- Todas sus entradas son visibles
- Todas sus salidas son visibles
- Para cada entrada siempre retornan la misma salida

Permite analizar su comportamiento de forma local y atemporal. Facilita:

- Reordenar el flujo de datos (Concurrencia, Optimización)
- Mover el flujo de datos (Sistemas Distribuidos)

Programación Funcional

Dada una función matemática

```
fibonacci :: Int -> Int
```

Se puede determinar:

- No solicita datos al usuario, internet, sistema de archivos, etc
- No accede al sistema de archivos
- No usa tarjeta de crédito para comprar cosas
- No lanza todos los misiles

Programación Funcional

Por cuestiones de marketing, lenguajes multiparadigma dicen soportar programación funcional cuando:

- Soportan funciones como componentes de primera clase: Pueden usarse funciones como variables, argumentos y retorno a funciones

- No requieren funciones matemáticas: No hay forma de verificar que una función sea matemática. Usualmente hay una convención social

Programación Logica

Basado en predicados:

```
fileExtension(Name, Ext) ; Se cumple si Name tiene extension Ext
```

Un programa es:

- Axiomas/Base de datos (X siempre se cumple)
- Reglas de inferencia (Si X, Y, Z se cumplen, entonces A se cumple)
- Consulta

Busca asignaciones de variables para deducir la consulta a partir de las reglas y axiomas

Programación Logica

Habíamos mencionado antes una semántica axiomática de sort:

- Pre: `lista` es una lista
- Post: `result` es permutación de lista
- Post: Si i, j son indices validos, $i \leq j \Leftrightarrow result[i] \leq result[j]$

Código prolog equivalente:

```
sort(Lista, ListaOrdenada) :-
    permutationOf(Lista, ListaOrdenada),
    ordered(ListaOrdenada).
```

Programación Logica

- Si queremos mejores resultados, el sistema necesita una mejor descripción de las características del problema
  - Considerar la clase de conocimiento usado para **demostrar** la correctitud del algoritmo
- Podemos obtener una solución con un mínimo de especificación, a costa de que el algoritmo usado es fuerza bruta
  - Precio de programador vs precio de cómputo → puede ser un muy buen trade-off
  - Al menos tenemos la opción

Modelo Relacional

Esquema de bases de datos, conceptualmente similar a programación lógica. Usualmente:

	Base de Datos	**Reglas**	**Turing-completo**
Logico	Pequeña	Muchas	Intencional
Relacional	Grande	Pocas	Workarounds

Modo de fallo común: Usar una herramienta optimizada de uno para el otro

Combinaciones de Paradigmas

- **Objetos**: Sistemas que se comunican por mensajes, encapsulando su estado
  Funcional: Dado un estado, determina predeciblemente el siguiente estado

➔ Patrón de Arquitectura: Functional Core, Imperative Shell
➔ Típico para sistemas distribuidos donde cada nodo debe conocer el estado de una simulación

Combinaciones de Paradigmas

- **Lógico**: Búsqueda de soluciones dentro de ciertas reglas
  Funcional: Analisis/manipulacion de flujo de datos

➔ Búsqueda eficiente dentro de estructuras simbólicas complejas

Combinaciones de Paradigmas

- **Funcional**: Prohibir comunicación por estado global mutable, para permite leer el grafo de uso de datos en los llamados a funciones
  Objetos: Encapsulamiento, el contrato se cumple sin importar cómo sucede por dentro

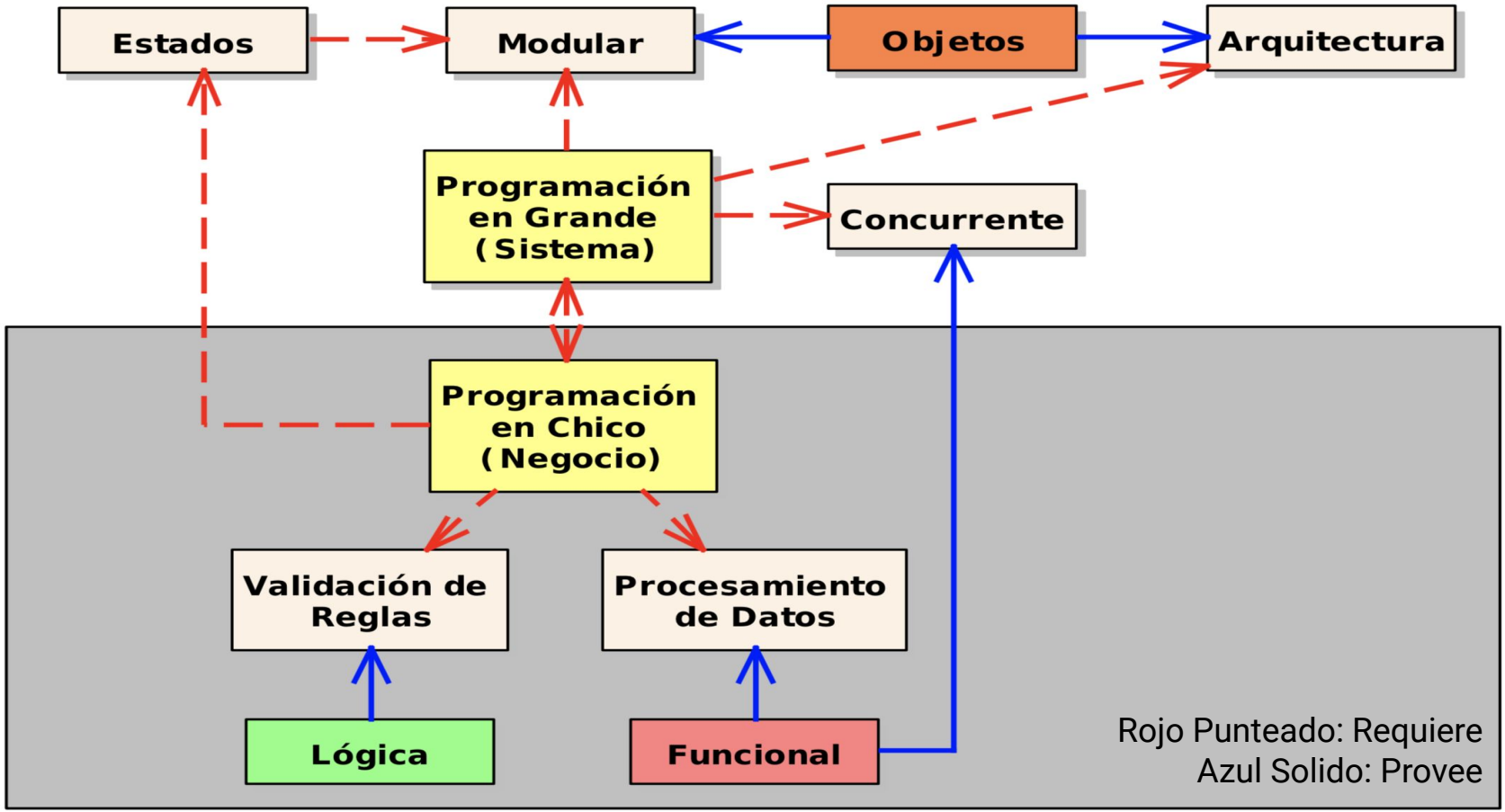
➔ Si necesitamos tener un grafo de uso de datos legible, se debe prohibir la lectura y mutación de estado global

Combinaciones de Paradigmas

- **Objetos**: Grafo de entidades, interactuando de a pares
  Relacional: Conjunto de hechos, interactuando en conjuntos

- **Objetos**: Concepto básico de herencia
  Relacional: Simulacro de herencia (Tabla por clase instanciable, tabla por clase con estado, tabla por árbol de clases?)

➔ Conflictos de representación: **Object–relational impedance mismatch**


Estados
Modular
Objetos
Arquitectura
Programación en Grande (Sistema)
Concurrente
Programación en Chico (Negocio)
Validación de Reglas
Procesamiento de Datos
Lógica
Funcional
Rojo Punteado: Requiere
Azul Solido: Provee