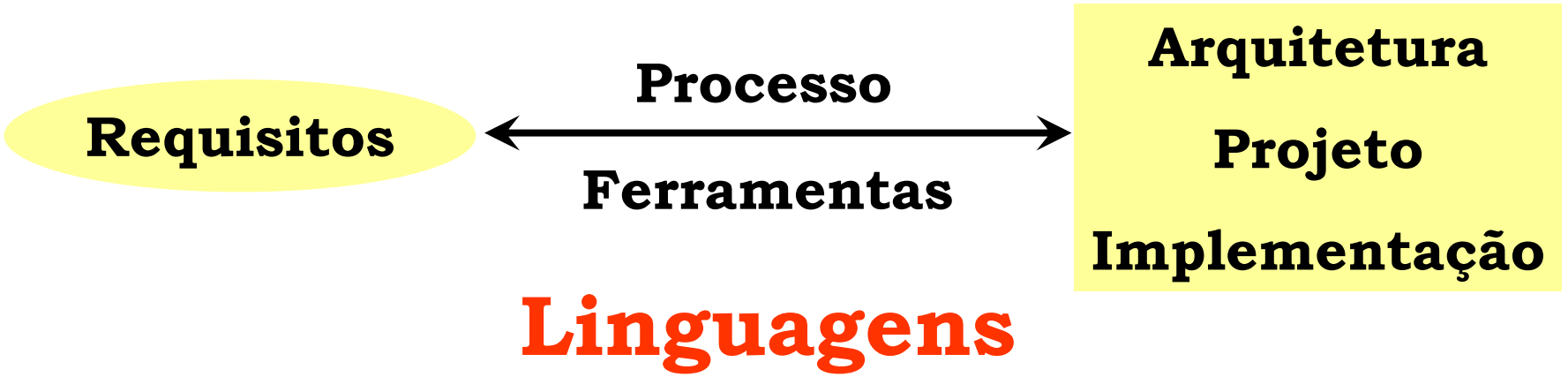
Construção de software e suas linguagens

Paulo Borba
Centro de Informática, UFPE

Construção de software



Linguagens influenciam como pensamos...

Language shapes the way we think, and determines what we can think about

Benjamin Lee Whorf

Linguista, 1897-1941

o que podemos ver e fazer...

The limits of my language mean the limits of my world

Ludwig Wittgenstein

Filósofo, Lógico, 1889-1951

e como programamos...

The use of COBOL criples the mind; its teaching should, therefore, be regarded as a criminal offence

Edsger Dijkstra

Cientista da Computação, 1930-2002

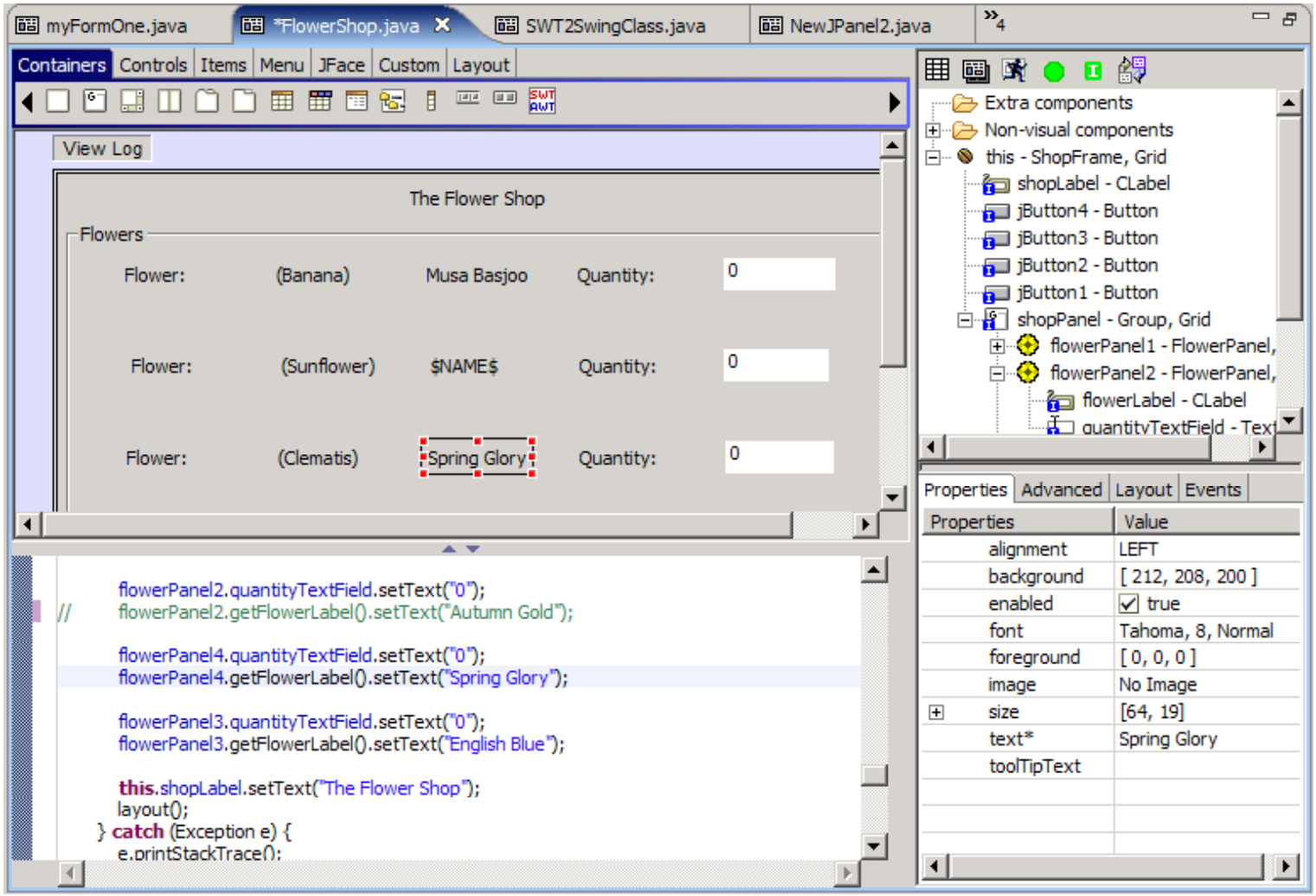
Linguagens...

- de Programação
- de Modelagem (Projeto, Arquitetura)
- Visuais
- de Domínio-específico
- ...
- Orientadas a objetos, fortemente tipadas, de metaprogramação, etc.

Linguagens de programação

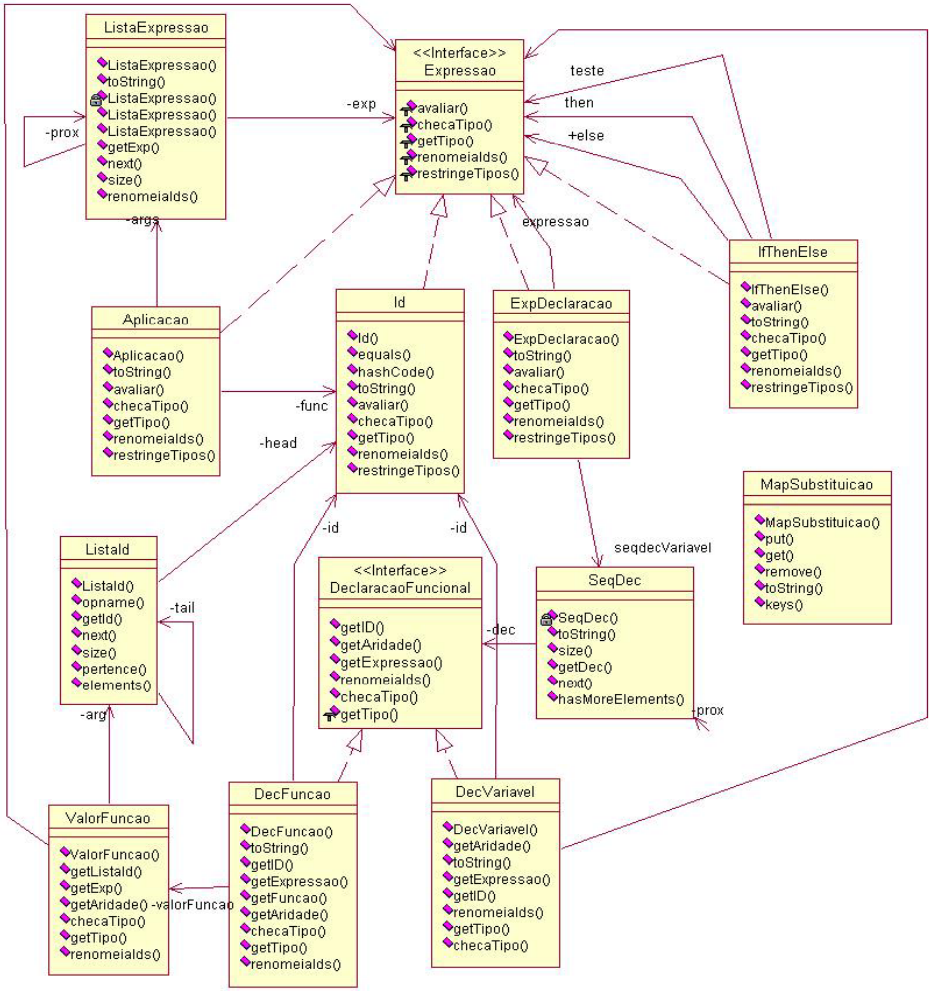
- Programa (código) é o artefato principal
  - Executável
  - Compiladores, interpretadores
  - Como fazer, não o que fazer
- Normalmente textuais
  - Mas programa também pode ser visualizado, e criado, graficamente

Programação visual de GUIs



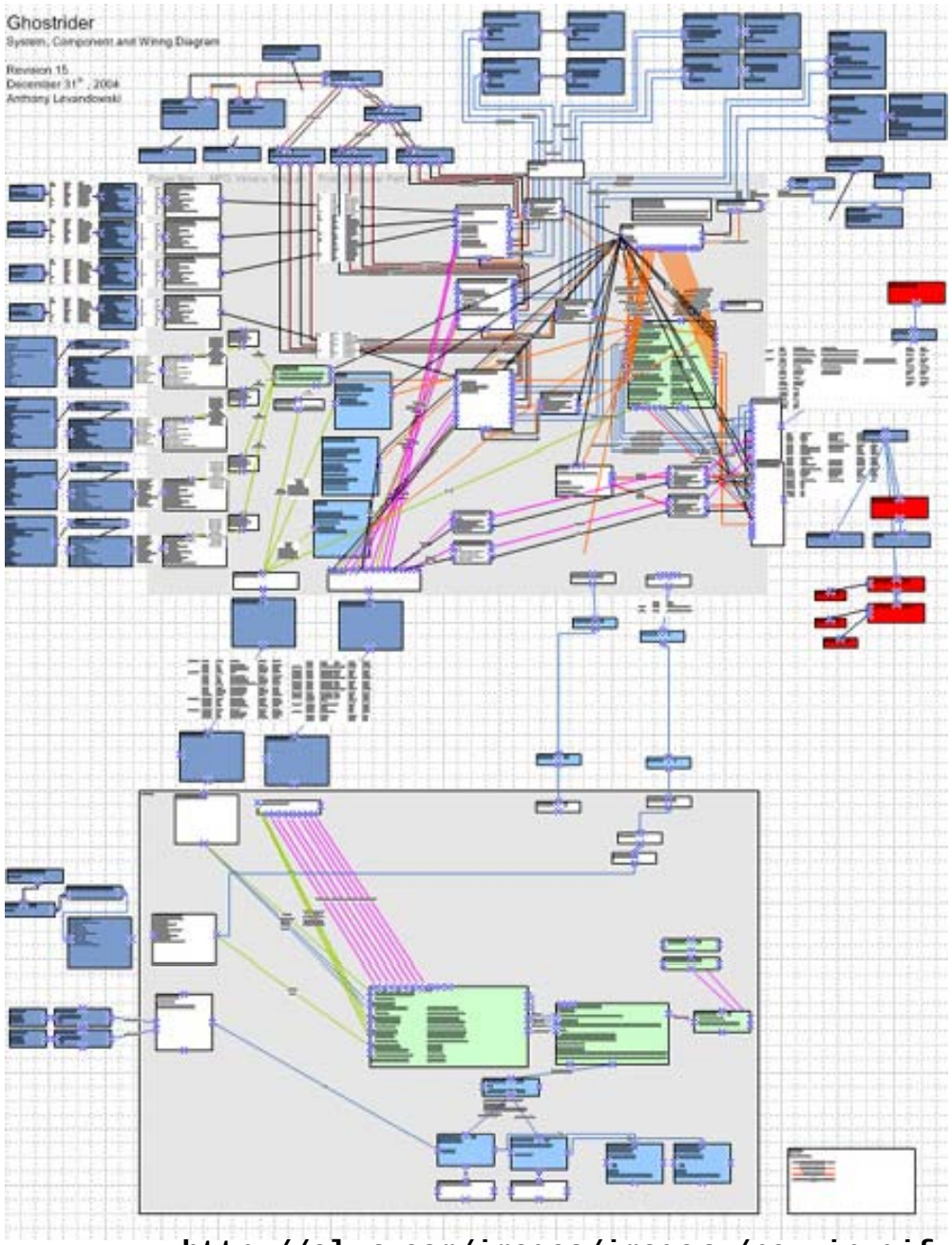
http://www.cloudgarden.com/jigloo/index.html

Programação visual com UML



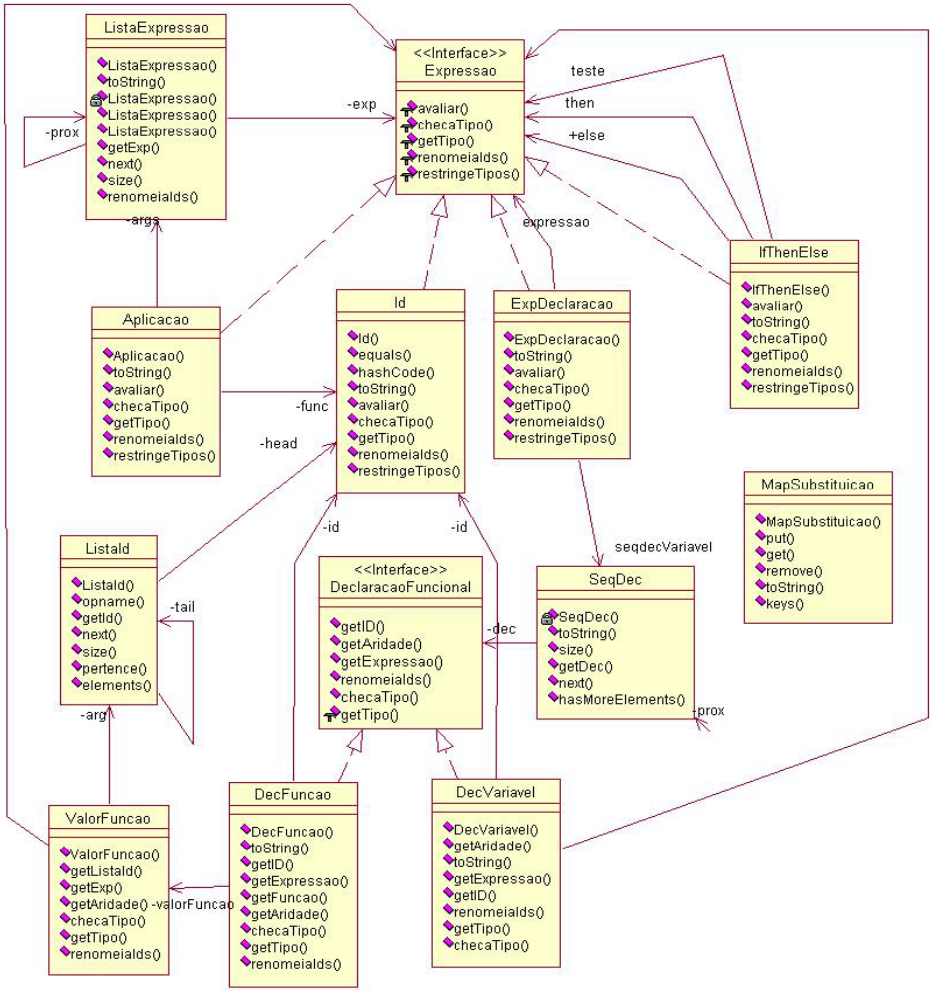
- UML para criação do programa
- MDA e UML executável
- Modelo ou programa?

- # Programa mesmo!
- Só um pouco mais de abstração em relação a um código textual
- Como fazer
- Executável
- Não tão útil para entendimento
- Geração também seria possível com tags no código textual



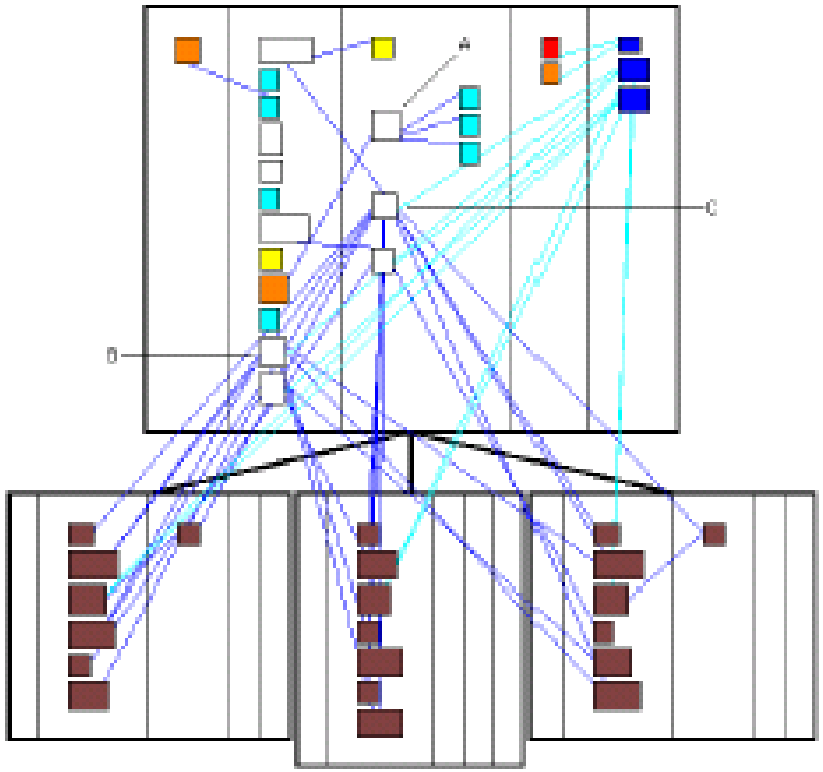
http://elzr.com/images/imagery/newwin.gif

Visualização de programas



- UML
- Engenharia reversa
- Modelo ou programa?

Temos também outras formas para visualizar classes…



http://www.iam.unibe.ch/~scg/Research/CodeCrawler/

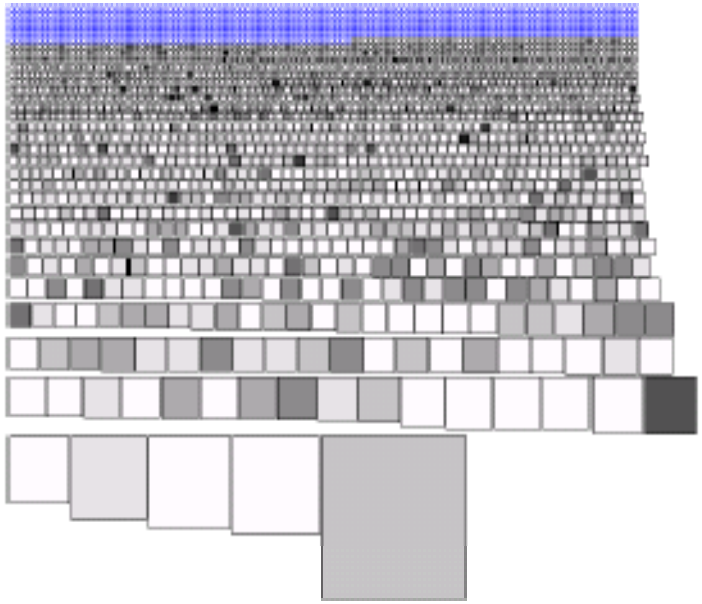
Class blueprint

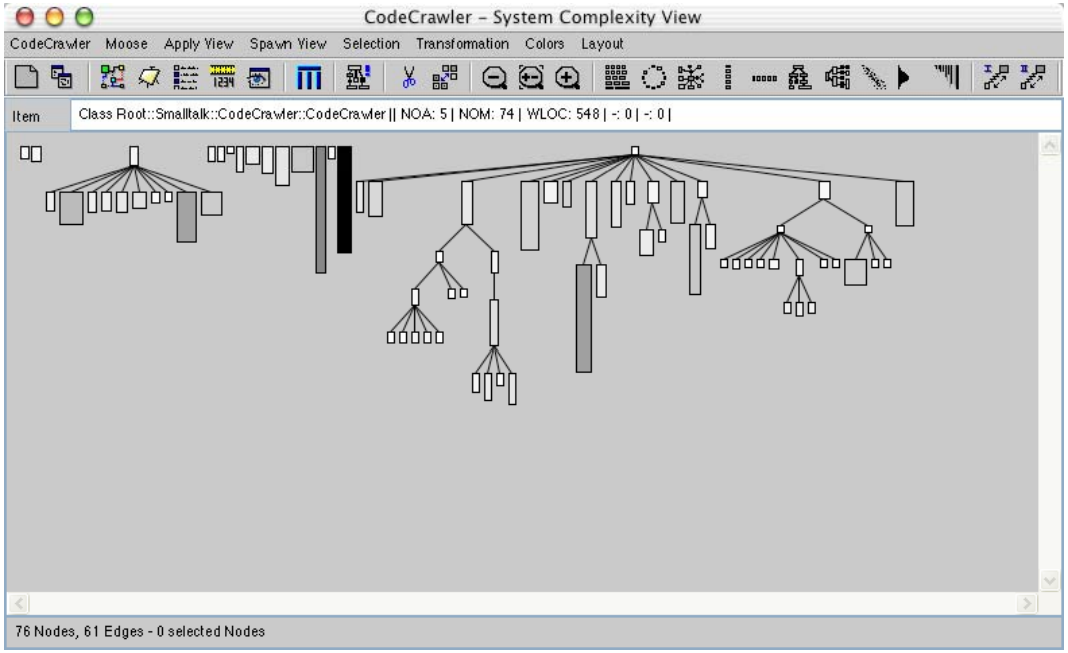
Description	Color
Attribute	blue node
Abstract method	cyan node
Extending method. A method which performs a *super* invocation.	orange node
Overriding method. A method redefinition *without* hidden method invocation.	brown node
Delegating method, delegates an invocation, *i.e.*, forwards the method call to another object.	yellow node
Constant method. A method which returns a *constant* value.	grey node
Interface and Implementation layer method.	white node
Accessor layer method. Getter.	red node
Accessor layer method. Setter.	orange node
Invocation of a method.	blue edge
Invocation of an accessor. Semantically equivalent to a direct access.	blue edge
Access to an attribute.	cyan edge

http://www.inf.unisi.ch/faculty/lanza/Downloads/Duca05b.pdf

milhões de linhas de código...

Quantidade de métodos por classe

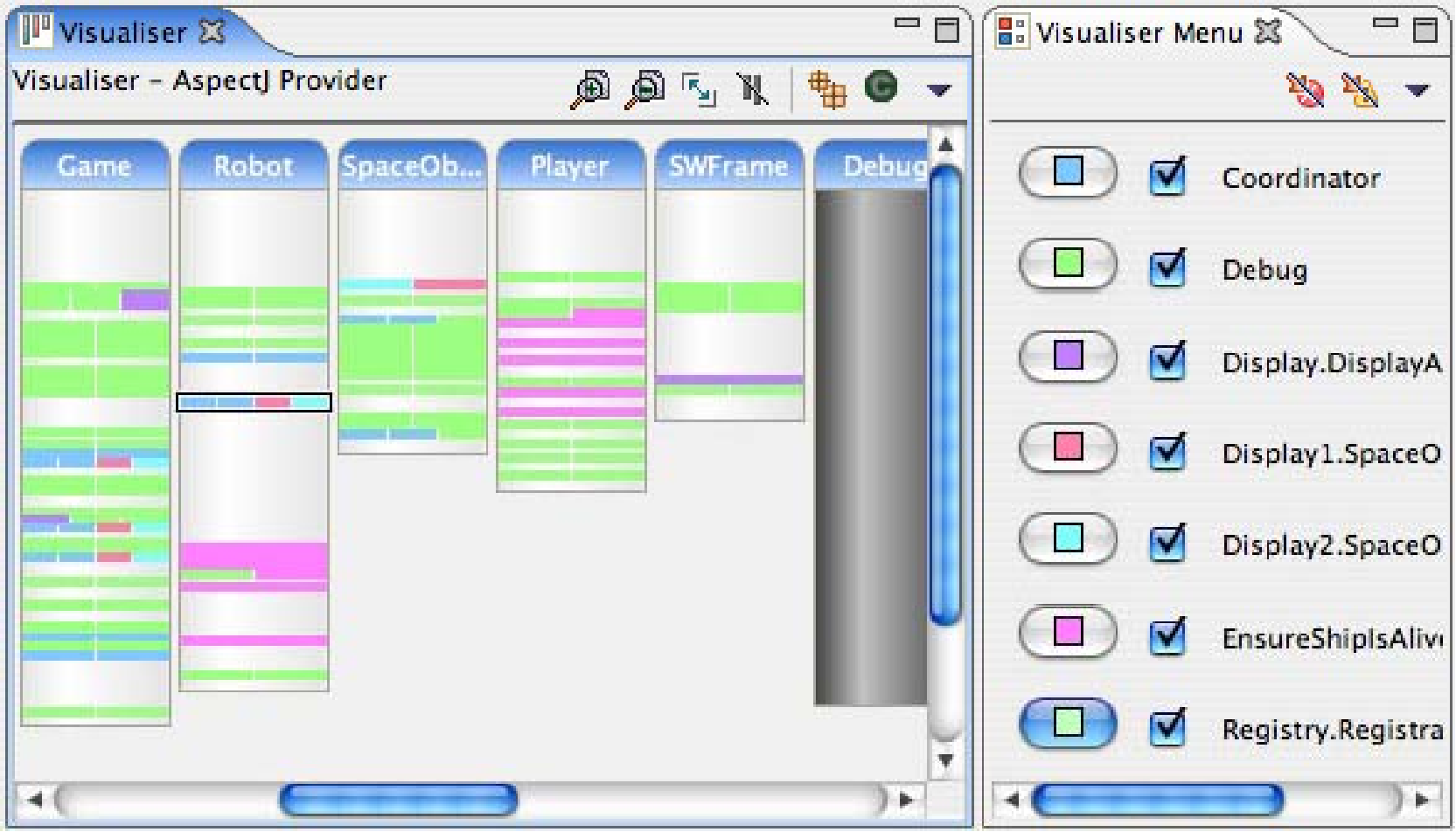




Quantidade de LOC, métodos e atributos por classe

http://www.iam.unibe.ch/~scg/Research/CodeCrawler/

e concerns, entre outras...



http://www-128.ibm.com/developerworks/java/library/j-aopwork9/

Programação visual agrega?

- Deveria no caso de GUIs...
- Não no caso de MDA
  - Geração e transformação, e maior abstração, é viável com texto também
  - Uso de ferramenta gráfica pode, na prática, atrapalhar

Isso não quer dizer que você não deva usar MDA!

Linguagens de modelagem

- "Modelo" (de projeto ou arquitetura) é o artefato principal
  - Não executável
  - O que fazer, e idéia geral de como fazer
  - Avaliação sem precisar construir o sistema
- Normalmente gráficas, mas podem ser textuais e, principalmente, híbridas

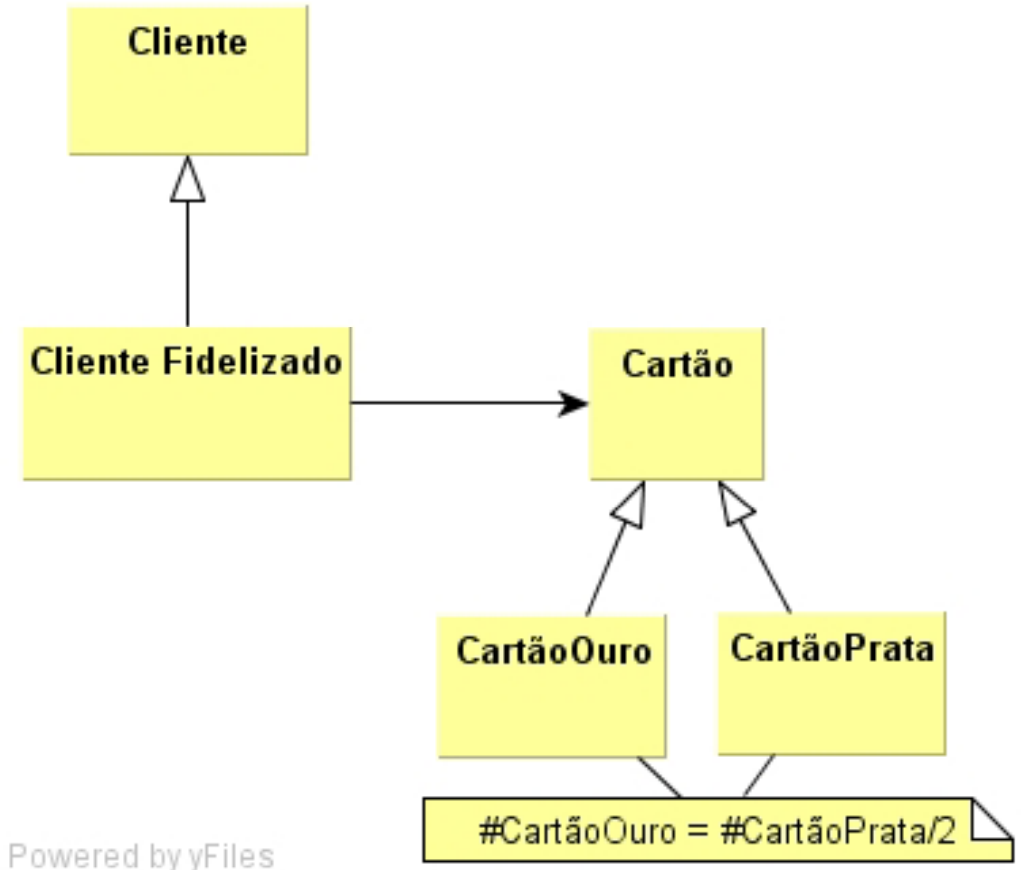
Modelos são úteis para...

- Criar entendimento antes de construir o software
- Possibilitar desenvolvimento composicional
- Facilitar comunicação e entendimento
- Avaliar, ajudar a tomar decisões

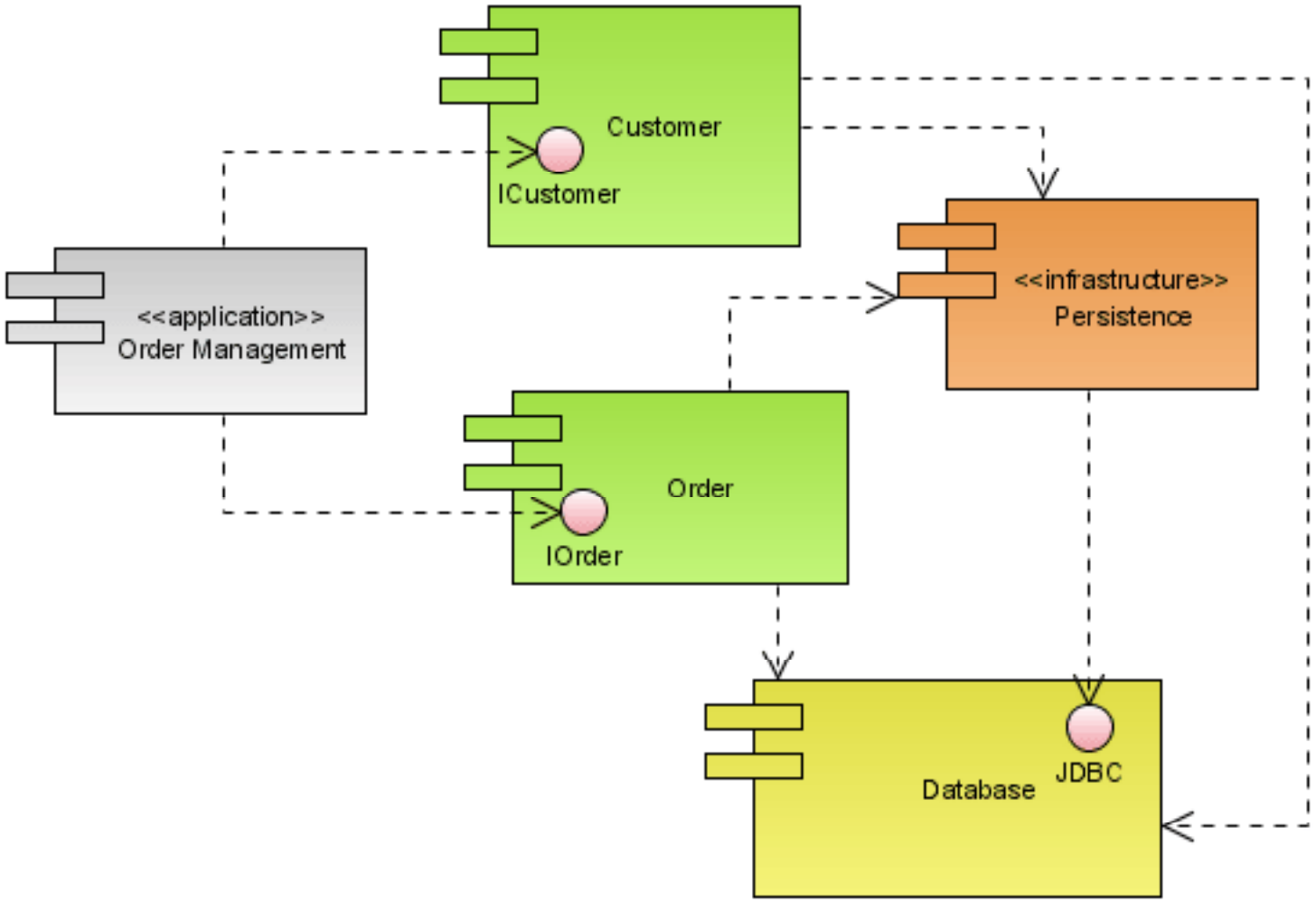
Mas foco deve ser em arquitetura

ou micromodelos (lightweight modeling)

Diagramas de classe abstratos



Diagramas de componentes



http://www.visual-paradigm.com/VPGallery/img/diagrams/Component/Component-Diagram-Sample.png

Linguagens visuais...

Acabamos de ver algumas... tanto de programação quanto de modelagem!

quando usar?

- Boas para definir
  - interconexão entre componentes
  - configuração de linhas de produtos
  - mas não para restrições associadas
- Razoáveis para definir
  - estados de um sistema
- Ruins para definir
  - consultas
  - fluxo de controle

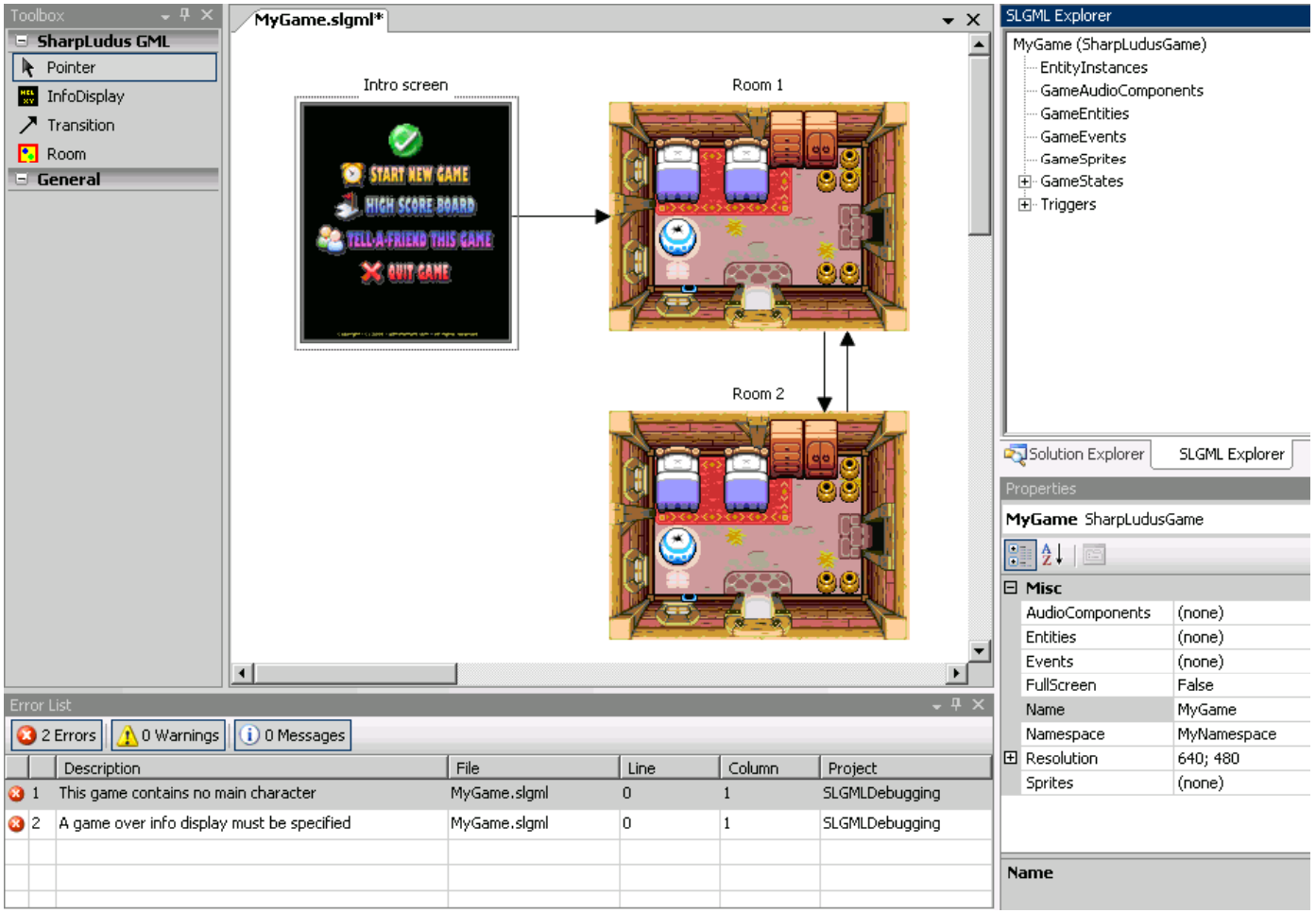
Linguagens de domínio específico

- Modelagem ou programação
- Envolvem apenas conceitos do domínio
- Reduz gap semântico entre problema e solução
- Alto nível de abstração
- Quando se paga?
  - Certamente não só para configuração

Domínios...

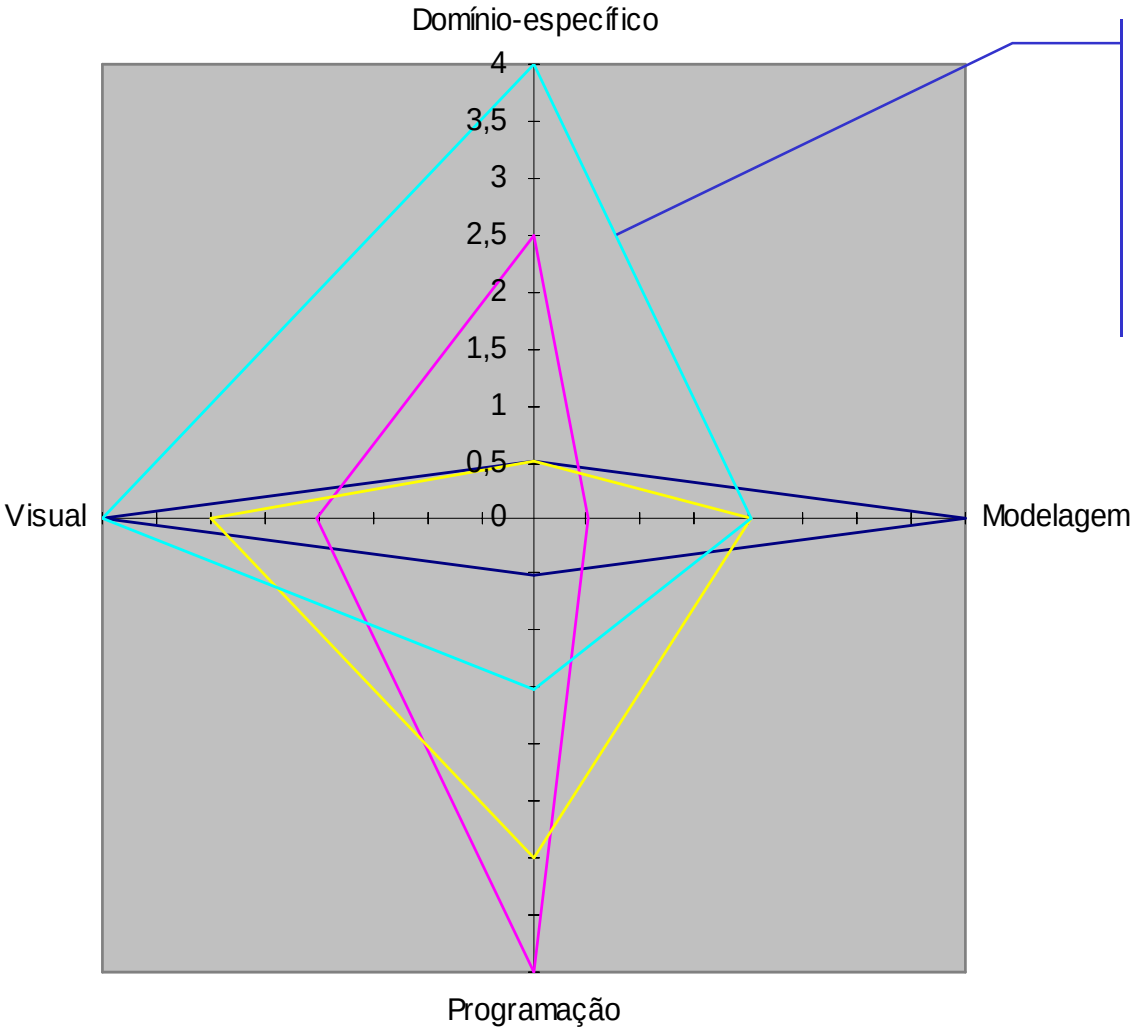
- Financeiro [conta corrente], jogos [de aventura], educação [controle acadêmico]...
- Aplicações web, variações em linhas de produto, configuração, acesso a dados...
- Não J2EE, .NET, Ciência da Computação...

Linguagem gráfica para jogos



http://www.cin.ufpe.br/~sharpludus

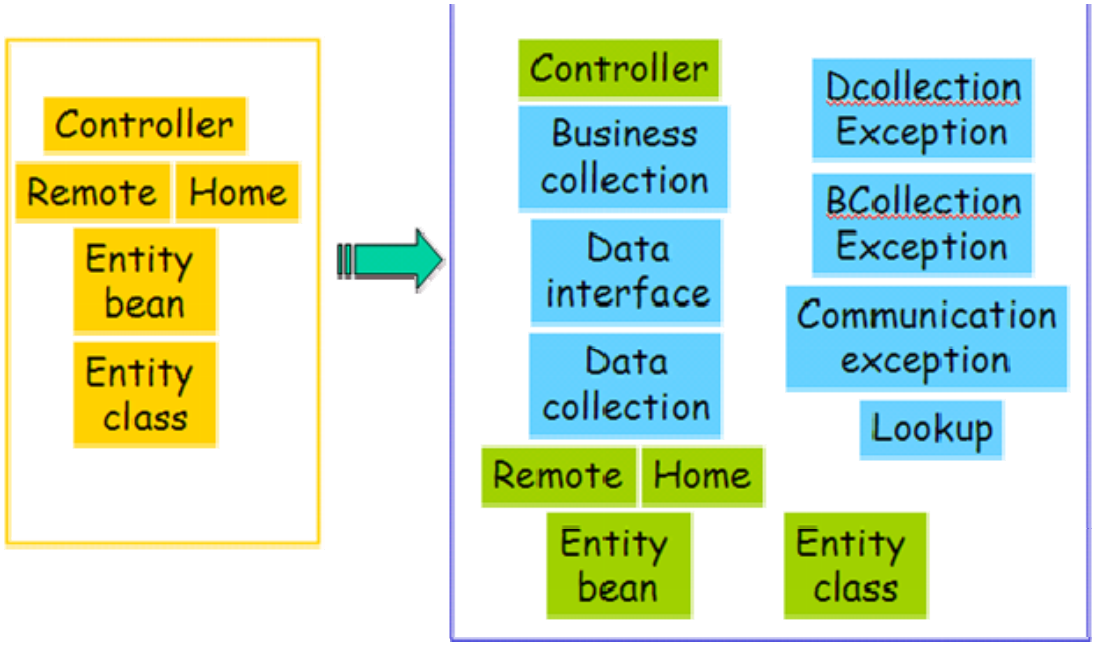
Abordagens e características de linguagens



Explora linhas de produtos de software, como forma de reduzir o esforço de programação

A implementação, ou uso prático, das linguagens requer...

- Geradores
- Transformadores



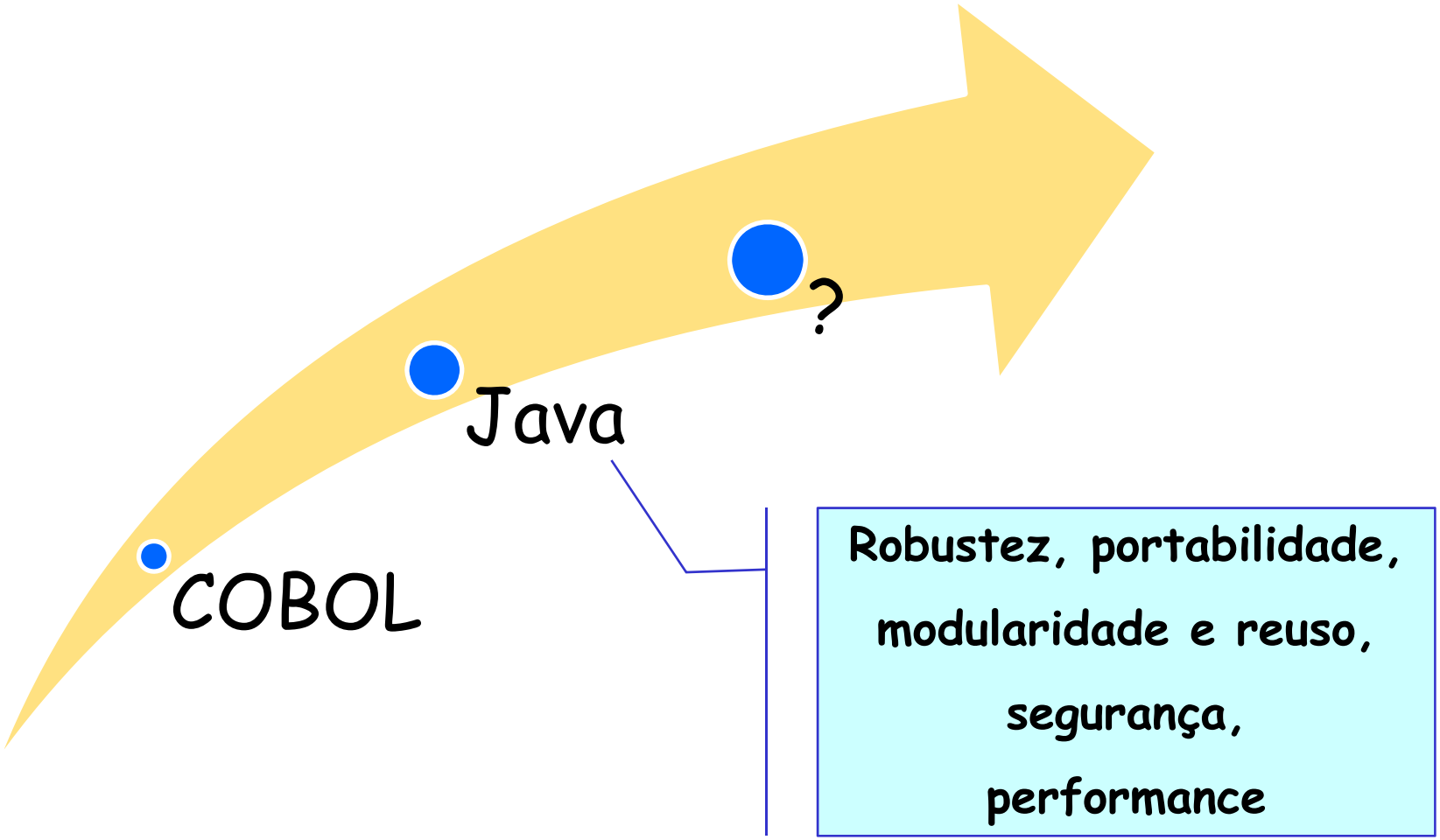
Transformadores e geradores para, além do tradicional,...

- Geração de código repetitivo
  - Coleções a partir de classe básica
  - Implementações de padrões de projeto
- Geração de código dependente de API
  - Independência de "plataforma"
  - Modelos dependentes de plataforma distorcem a própria idéia de modelo
- Refactoring

Transformadores e geradores…

- Como parte da implementação da linguagem
  - UML e MDA
- Como ferramentas a parte
  - Provavelmente a linguagem precisa evoluir, ter extensões para um determinado domínio
  - Hibernate e java.sql versus Linq
  - Cada API uma linguagem…

Evolução das linguagens



Próxima Java...

- AspectJ? Ruby? Fortress? Scala? ...
- Em quanto tempo?
- Com que características?

Características prováveis

- Sintaxe mínima
- Muito mais abstração e reflexão
- Concorrência, persistência...
- Funções de alta ordem
- Aspectos, mixins, traits
- Componentes, configuração
- Growable (sintaxe e semântica), DSLs
- Inferência de tipos

Linguagens de script

- Integração com recursos (DSLs) para
  - Processamento de strings (expressões regulares)
  - Manipulação de processos
- Sintaxe mínima e flexível
  - Operadores infixos
  - Evita declarações de tipos, interfaces
- Reflexão computacional
  - Com a sintaxe da linguagem
  - Metaprogramação
  - Mixins
- Alta-ordem e closures
- Estruturas de dados de alto nível
  - Map, bag, set, list

Estruturas mais abstratas

```
a = [“zero”, “um”, “dois”, “três”]
b = a – [“um”]
c = {“zero” => 0, “um” => 1}
puts c[“dois”]
c.delete(“um”)
a = a + c.values
```

http://www.pragmaticprogrammer.com/titles/bmsft

Sintaxe mínima, DSLs...

```
def asterisks_for(an_integer)
  '*' * (an_integer / 5.0).round
end


`svn log #{timespan} #{root}/#{subsystem}`


/\((\d+)\)/.match(line)[1].to_i
```

http://www.pragmaticprogrammer.com/titles/bmsft

Sintaxe mínima e flexível

```
class NoteTaker
  attr_reader :commentary
  def initialize(title, extra=“ “)
    @commentary = [title << extra]
  end
  def note(notation)
    @commentary << "Note: #{notation}"
  end
end
```

http://www.pragmaticprogrammer.com/titles/bmsft

Sintaxe mínima e adequada ao domínio...

ASCII	UNICODE	Two-dimensional
`rho0 = r DOT r`	$\rho_0 = r \cdot r$	$\rho_0 = r \cdot r$
`v_norm = v / norm v`	$v\_norm = v / \|v\|$	$v_{norm} = \frac{v}{\|v\|}$
`SUM[k=1:n] a[k] x^k`	$\Sigma[k=1:n]\ a[k]\ x^k$	$\sum_{k=1}^{n} a_k x^k$
`C = A UNION B`	$C = A \cup B$	$C = A \cup B$

http://research.sun.com/projects/plrg/Publications/OOPSLA-Growable-Language-2006public.pdf

muito adequada ao domínio!

```
object SolarSystem extends { StarSystem, OrbitingObject }
    sun = Sol
    planets = { Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune }
    position = Polar(25000 lightYears, 0 radians)
    ω : ℝ64 AngularVelocity = 2π radians / 226 million years in seconds

    variation(ω_Δ) =
        ω += ω_Δ
end
```

```
object Particle(position : (ℝ Length)^3,
                velocity : (ℝ Velocity)^3)
    extends Moving
    getter velocity() = do
        print "velocity getter accessed"
        self.velocity
    end
end
```

http://research.sun.com/projects/plrg/Publications/fortress1.0beta.pdf

Mais flexibilidade e abstração

```
printFirst(xs : ℝ ...) =
    if |xs| > 0 then print xs_0
    else throw Error end
```

```
factorial(n)
    requires { n ≥ 0 }
    ensures { result ≥ 0 }
    = if n = 0 then 1
      else n factorial(n − 1) end
```

```
case planet ∈ of
    { Mercury, Venus, Earth, Mars } ⇒ "inner"
    { Jupiter, Saturn, Uranus, Neptune } ⇒ "outer"
    else ⇒ "remote"
end
```

http://research.sun.com/projects/plrg/Publications/fortress1.0beta.pdf

Iterators? Map e filter!

```
def inventory_from(filename)
  inventory = File.open(filename)
  downcased = inventory.collect do | line |
    line.chomp.downcase
  end
  downcased.reject do | line |
    boring?(line)
  end
end
```

http://www.pragmaticprogrammer.com/titles/bmsft

Blocos como parâmetros

```
def simple
  yield
  puts 'block done.'
end
def with_arg(arg)
  yield arg
end
```

http://www.pragmaticprogrammer.com/titles/bmsft

User-defined iterators…

$$factorial(n) = \prod_{i \leftarrow 1:n} i$$

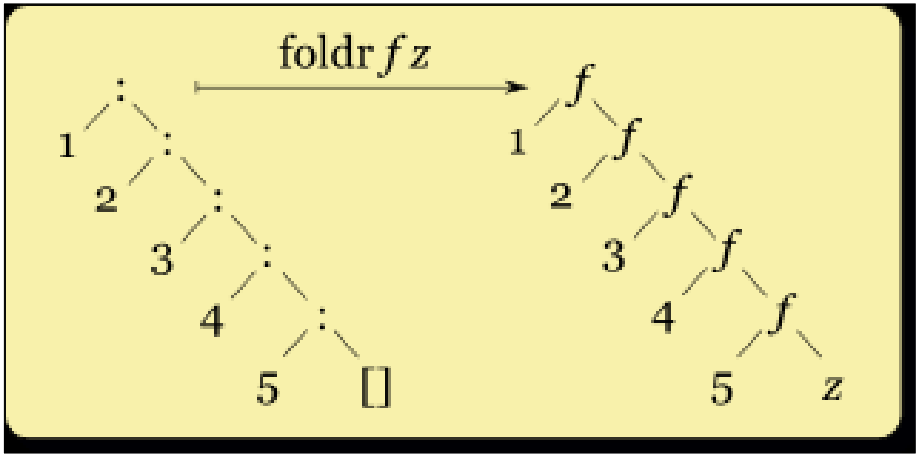
$$y = \sum_{k \leftarrow 1:n} a_k x^k$$

```
for i ← 1:10 do
    print(i " ")
end
```

$$\{x^2 \mapsto x^3 \mid x \leftarrow \{0,1,2,3,4,5\},\ x \,\texttt{MOD}\, 2 = 0\}$$

http://research.sun.com/projects/plrg/Publications/fortress1.0beta.pdf

baseados em catamorfismos




```
foldr f z [] = z
foldr f z (x:xs) =
      f x (foldr f z xs)
```

```
sum = foldr (+) 0
map f = foldr ((:).f) []
filter p = foldr (\y -> \ys -> if p y then y:ys else ys)  []
```

http://en.wikipedia.org/wiki/Fold_%28higher-order_function%29

Concorrência abstrata

```
do
   factorial(100)
also do
   factorial(500)
also do
   factorial(1000)
end
```

```
atomic do
    x += 1
    y += 1
end
```

http://research.sun.com/projects/plrg/Publications/fortress1.0beta.pdf

Queries integradas e unificadas

```
IEnumerable<string> query = from s in names
                            where s.Length == 5
                            orderby s
                            select s.ToUpper();
```

Ortogonal a ORM

Acesso a BDs, collections, XML, etc.

Segurança de tipos

http://msdn2.microsoft.com/library/bb308959.aspx

Metaprogramação

- Transformadores e geradores como parte da linguagem
  - Aspectos em AspectJ
  - Extensões em Ruby

Traits: herança múltipla de código, contratos, e testes

```
trait Any extends {}
   opr ≡(self, other: Any): Boolean
   opr IDENTITY(self): Any
   hash(maxval: ℕ64): ℕ64
   hash(maxval: ℕ32): ℕ32
   getter hashCode(): ℕ64
   toString(): String
   property ∀(x, y, n: ℕ64) x ≡ y → x.hash(n) ≡ y.hash(n)
   property ∀(x, y, n: ℕ32) x ≡ y → x.hash(n) ≡ y.hash(n)
   property ∀(x) x.hashCode ≡ x.hash(2^64 − 1)
   property ∀(x, y) x ≡ y → x.toString() = y.toString()
   property ∀(a) (IDENTITY a) ≡ a
   property ∀(a) a ≡ a
   property ∀(a, b) a ≡ b ↔ b ≡ a
   property ∀(a, b, c) (a ≡ b∧: b ≡ c) →: a ≡ c
end
```

http://research.sun.com/projects/plrg/Publications/fortress1.0beta.pdf

Mixins

```
module SomeModule
  class SomeClass …
  end
  Constant = "I am …"
  def self.speak
    "I am a module method."
  end
  def speak
    "I am a mixin method."
  end
end
```

```
class Threeer
  include Enumerable
  def each
    yield(1)
    yield(2)
    yield(3)
  end
end
```

http://www.pragmaticprogrammer.com/titles/bmsft

Aspectos, não invasão

```
public aspect PersistenceAspect {
  before(): transMethods() {
    getPM().startTransaction();
  }
  after() returning(): transMethods() {
    getPM().commitTransaction();
  }
  after() throwing(): transMethods() {
    getPM().rollBackTransaction();
  }...
```

Aspectos, quantificação

```
pointcut transMethods():
  execution(public * Trans.*(..));

private interface Trans {
  public void register(Account a);...
}
declare parents: Bank implements Trans;
```

Evolução de linguagens e do quicksort

Java [editar]

```
import java.util.Comparator;
import java.util.Random;

public class Quicksort {
    public static final Random RND = new Random();
    private static void swap(Object[] array, int i, int j) {
        Object tmp = array[i];
        array[i] = array[j];
        array[j] = tmp;
    }
    private int partition(Object[] array, int begin, int end, Comparator cmp) {
        int index = begin + RND.nextInt(end - begin + 1);
        Object pivot = array[index];
        swap(array, index, end);
        for (int i = index = begin; i < end; ++ i) {
            if (cmp.compare(array[i], pivot) <= 0) {
                swap(array, index++, i);
        }   }
        swap(array, index, end);
        return (index);
    }
    private void qsort(Object[] array, int begin, int end, Comparator cmp) {
        if (end > begin) {
            int index = partition(array, begin, end, cmp);
            qsort(array, begin, index - 1, cmp);
            qsort(array, index + 1,  end,  cmp);
    }   }
    public void sort(Object[] array, Comparator cmp) {
        qsort(array, 0, array.length - 1, cmp);
    }
}
```

Haskell

```
  sort :: (Ord a)   => [a] -> [a]

  sort []           = []
  sort (pivot:rest) = (sort [y | y <- rest, y < pivot])
                       ++ [pivot] ++
                      (sort [y | y <- rest, y >=pivot])
```

Lisp

```
(defun partition (fun array)
  (list (remove-if-not fun array) (remove-if fun array)))

(defun sort (array)
  (if (null array) nil
    (let ((part (partition (lambda (x) (< x (car array))) (cdr array))))
      (append (sort (car part)) (cons (car array) (sort (cadr part)))))))
```

Ruby

```
def sort( array )
   return array if array.size <= 1
   pivot = array[0]
   return sort( array.select { |y| y < pivot } )
          + array.select { |y| y == pivot } +
          sort( array.select { |y| y > pivot } )
end
```

http://pt.wikipedia.org/wiki/Quicksort

Quicksort inspirador

(Silvio Meira, 1983)

$$\text{sort}\ () = ()$$

$$\text{sort}\ (a : x) = \text{sort}\ \{b \leftarrow x;\ b \leq a\ \} +\!\!+ a : \text{sort}\ \{\ b \leftarrow x;\ b > a\}$$

http://www.cs.mdx.ac.uk/staffpages/dat/saslman.pdf

Concluindo...

Learn at least one new language every year. Different languages solve the same problems in different ways. By learning several different approaches, you can help broaden your thinking and avoid getting stuck in a rut.

The Pragmatic Programmer

Construção de software e suas linguagens

Paulo Borba
phmb@cin.ufpe.br
www.cin.ufpe.br/spg