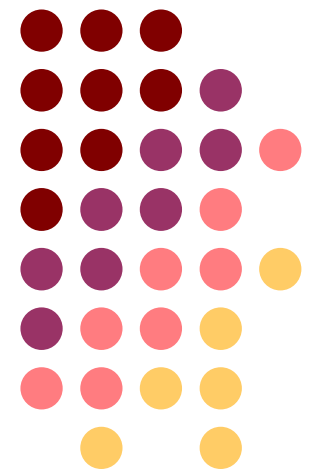


Linguagens de Domínio Específico: o que são e quando e como utilizar

André Santos
André Furtado

CIn-UFPE





Problema para resolver:

- Em um arquivo texto, encontrar endereços de email (user@host) e substituir o host por “cin.ufpe.br”.
- Que linguagem utilizar para resolver este problema?

Solução: usar expressões regulares



- Ferramenta (linguagem) sed

```
sed '/@[^ ]*/s//@cin.ufpe.br/g' < textfile
```

- Usar linguagem de expressões regulares embutida em outras linguagens ou via API (C#, Java etc.)



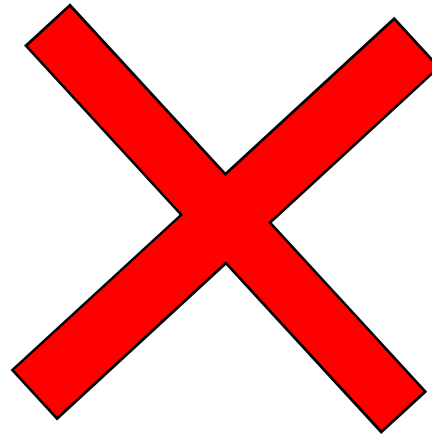
Outros Exemplos:

- Realizar uma consulta em um banco de dados
- Renomear todos os arquivos com extensão “.htm” para a extensão “.html”
- Calcular a média e desvio padrão de uma série de 100 números

Quais as diferenças de cada abordagem?



General Purpose
Languages

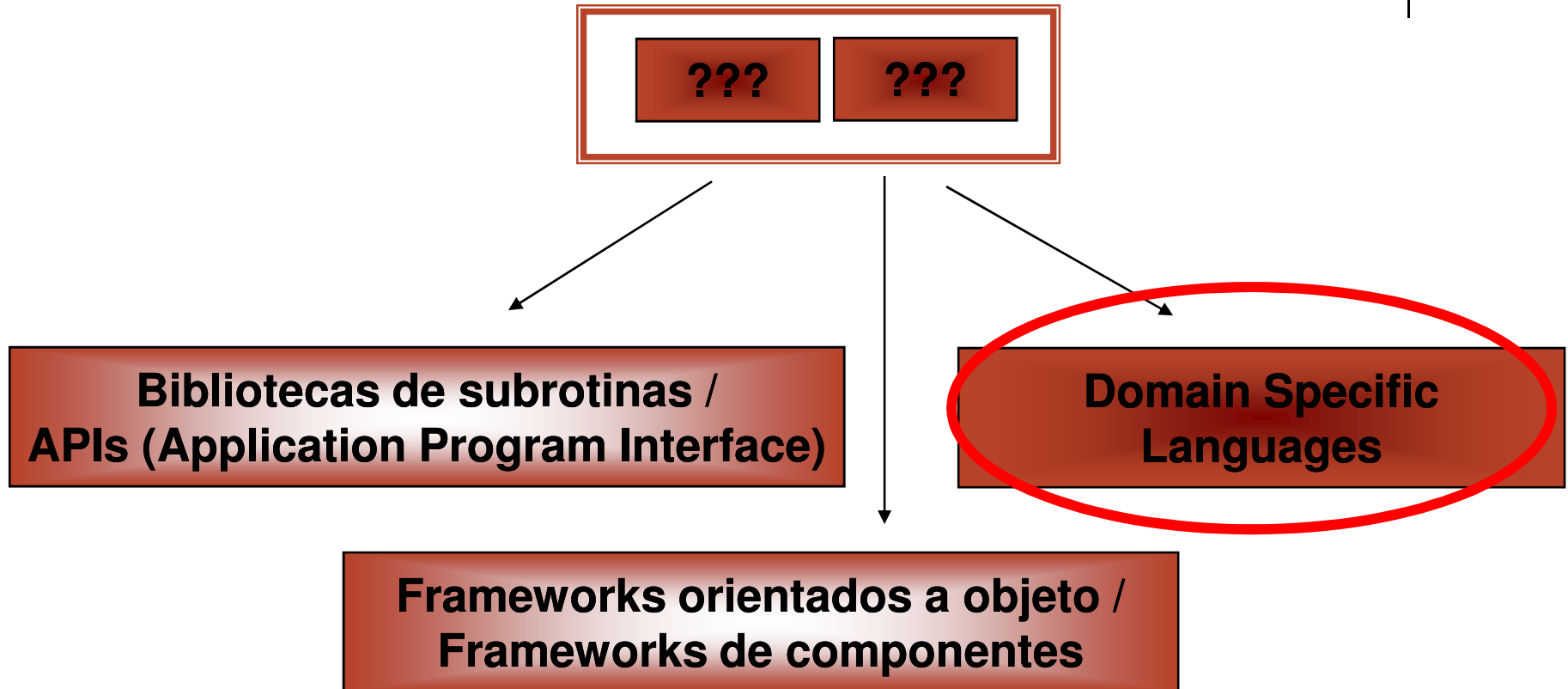


Domain-Specific
Languages

Um pouco de história...



Endereçando domínios específicos



Domain Specific Languages



- Vários problemas de desenvolvimento de software podem ser resolvidos mais facilmente através de uma linguagem com um propósito específico.

Domain Specific Languages



Linguagem que oferece, através de notações e abstrações adequadas, um poder de expressão focado em (e geralmente restrito a) um domínio particular

Características de DSLs



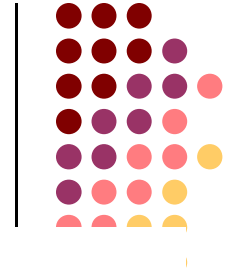
- Geralmente são pequenas
 - “Micro-languages” ou “little languages”
- Geralmente são declarativas
 - Podem ser vistas não apenas como linguagens de programação como também linguagens de especificação
- Muitas DSLs são traduzidas (compiladas) em chamadas a APIs

Exemplos de DSLs

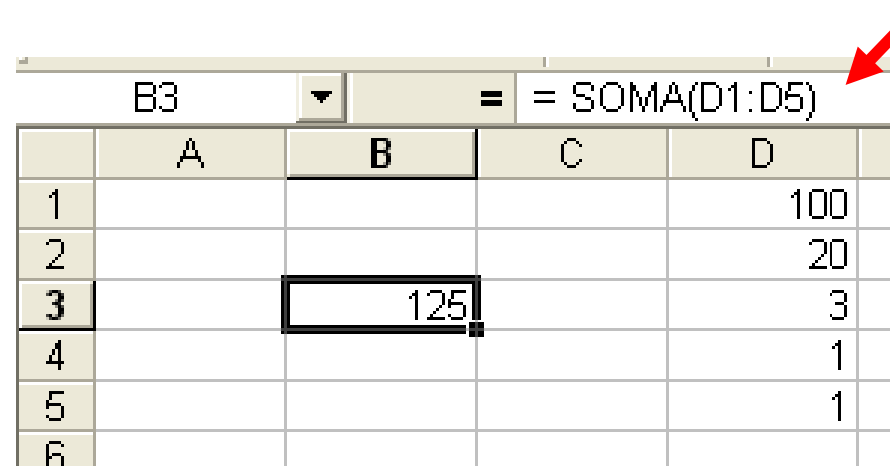


- **LEX**
- **YACC**
- **SQL**
- **BNF**
- **HTML**
- **JavaDoc**
- **TeX / LaTeX**
- **Linguagens para comandos na shell do sistema operacional**

Exemplos de DSLs

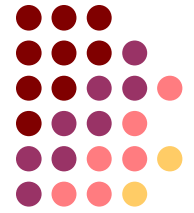


- DSLs estão em todos os lugares
 - A própria linguagem de fórmulas do Excel é uma DSL!
 - End-user programming

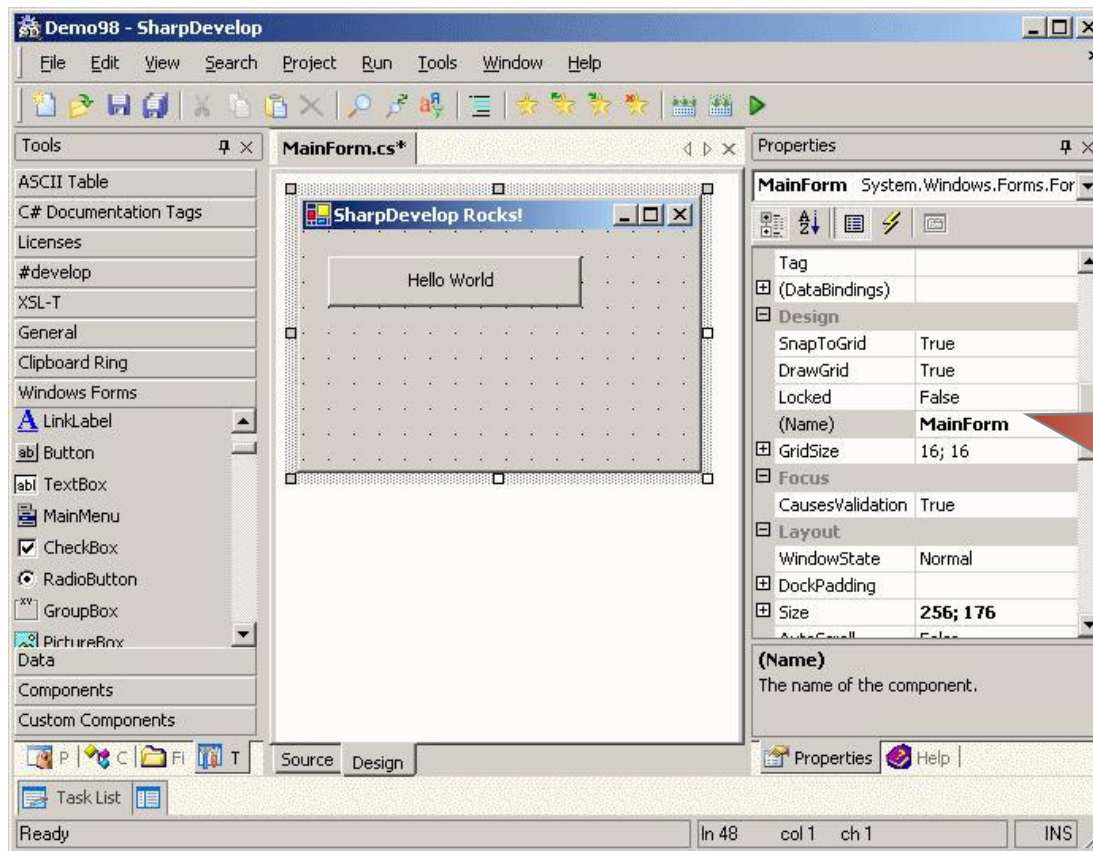


	A	B	C	D
1				100
2				20
3		125		3
4				1
5				1
6				

DSL Visuais



- Construtores de GUI também podem ser vistos como DSLs?

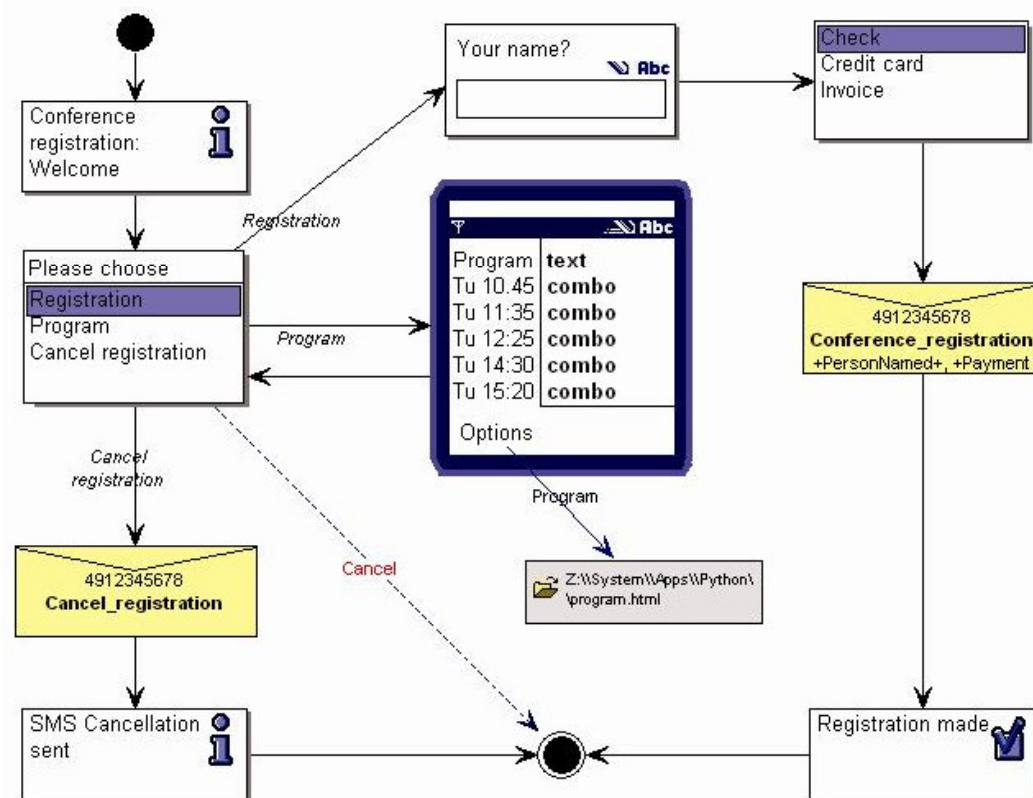


Sim, mas a experiência do usuário, entretanto, é diferente (pois a DSL é visual!)

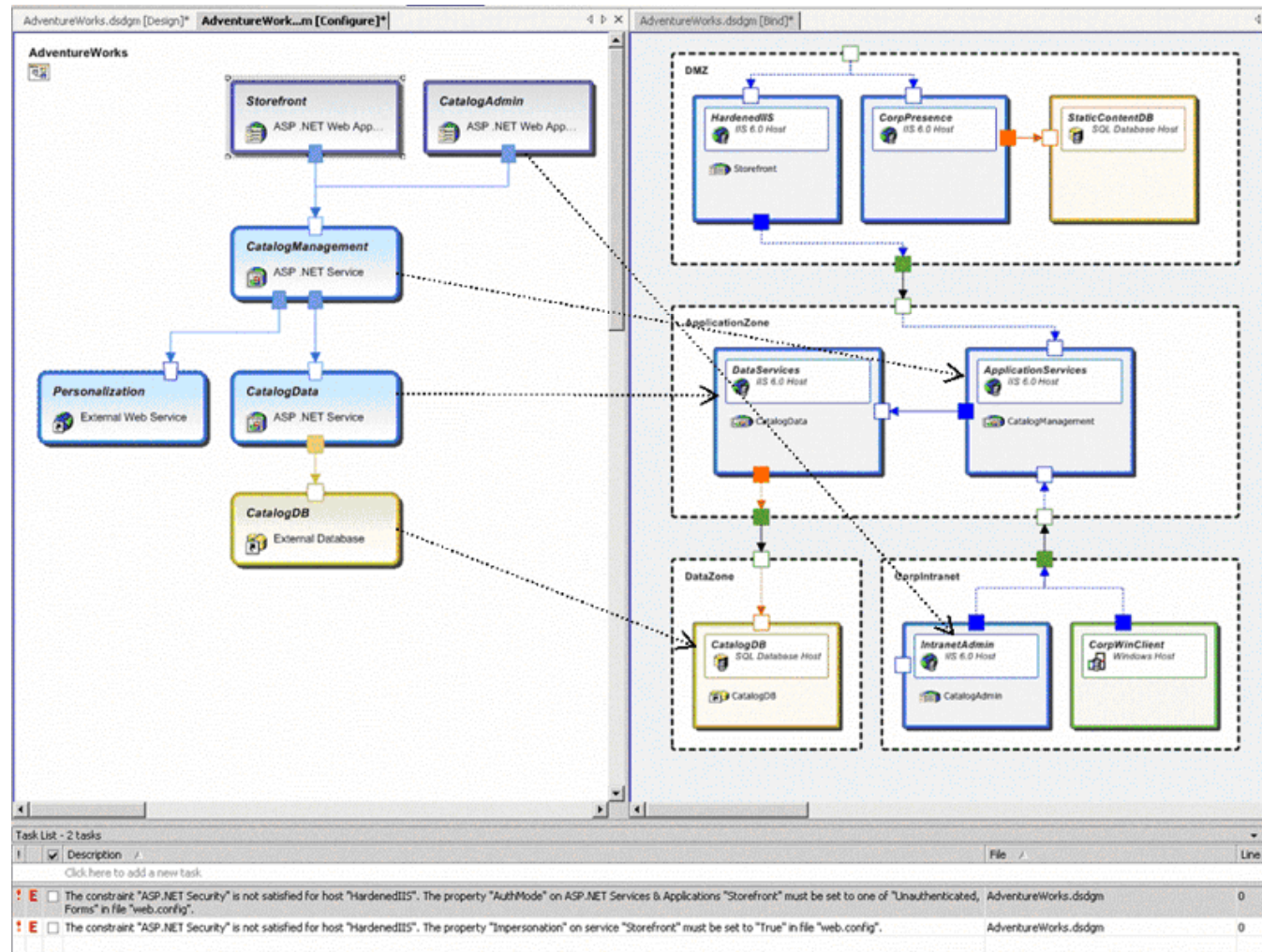
Modelagem visual



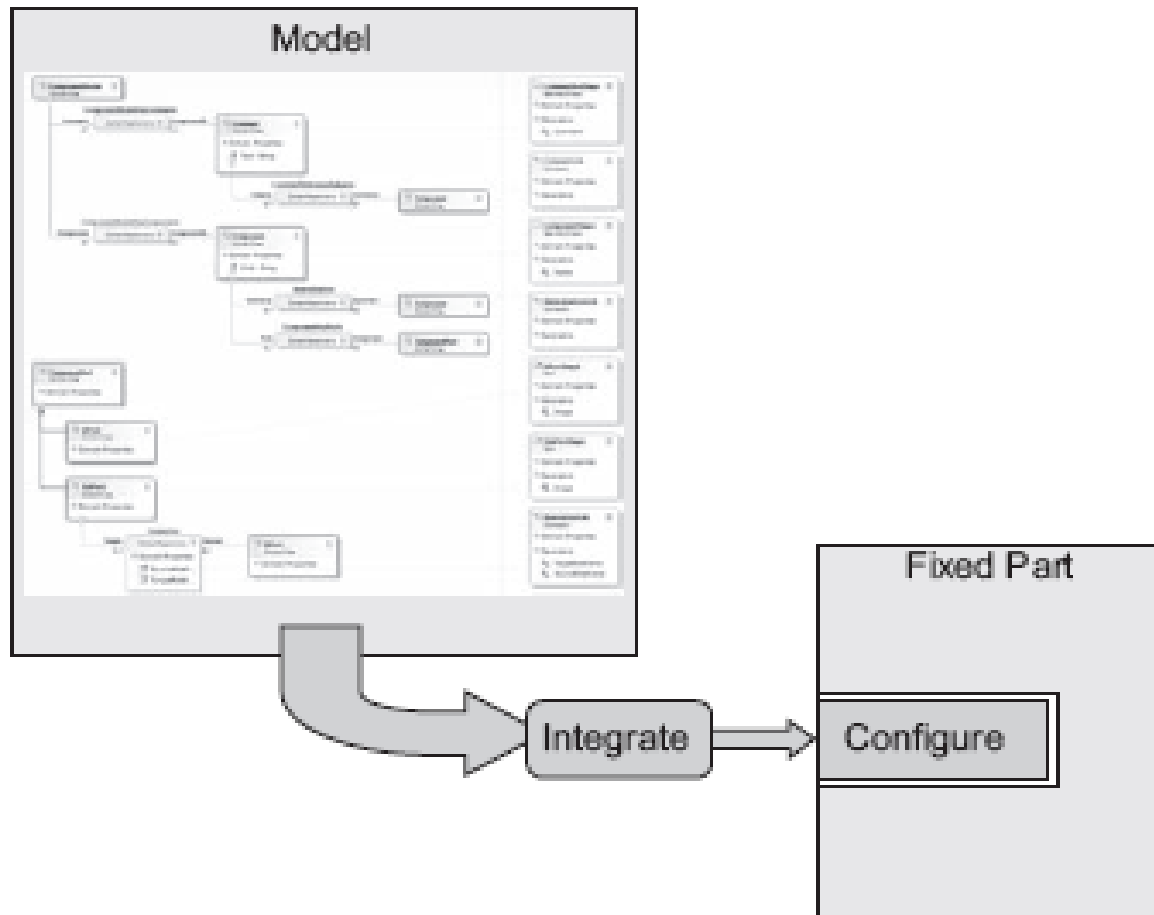
- Utiliza DSLs visuais para modelar a solução em um domínio
- Exemplo: linguagem para aplicações de smart phones



Outro Exemplo: VS Team System for Architects



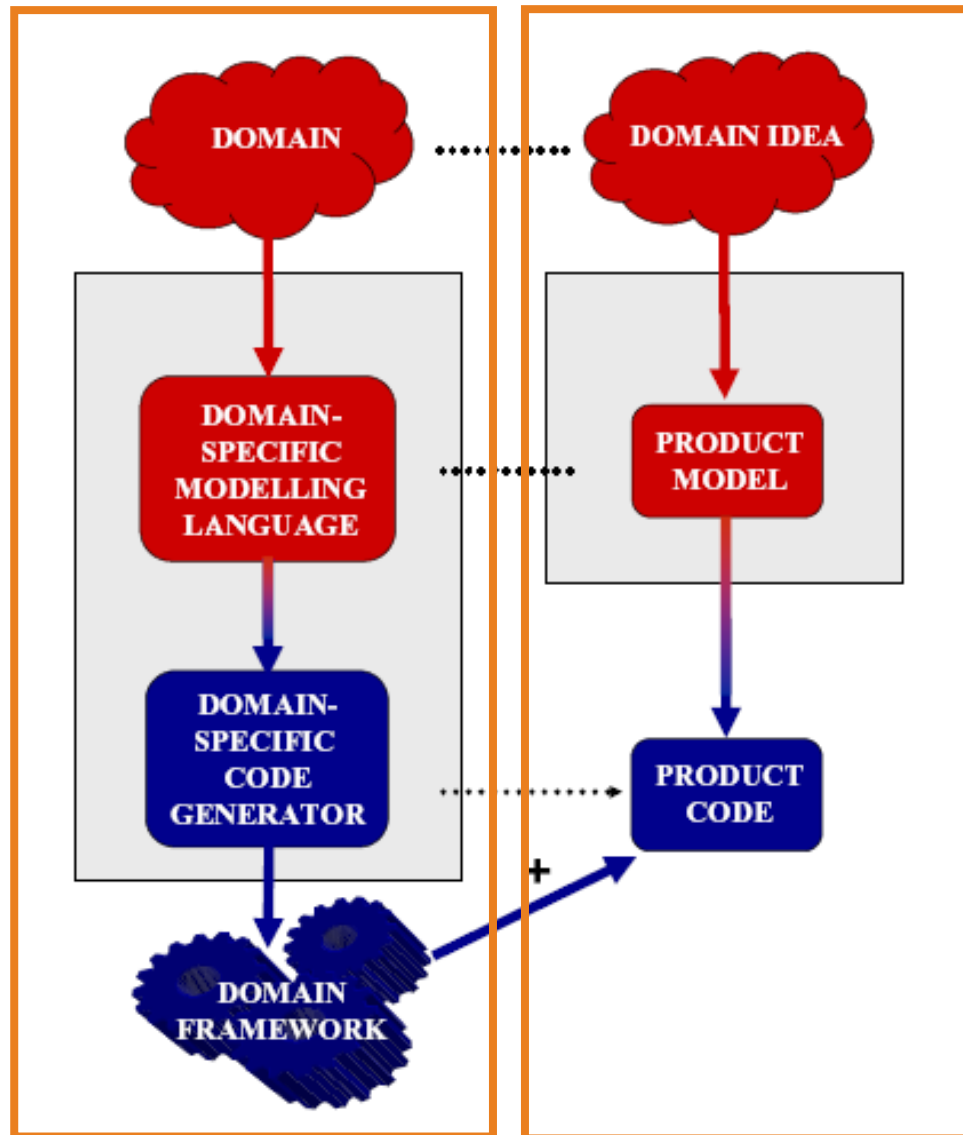
Domain Specific Development



Modelagem Visual



Feito apenas uma vez, pelo projetista da linguagem



Feito várias vezes, pelos desenvolvedores

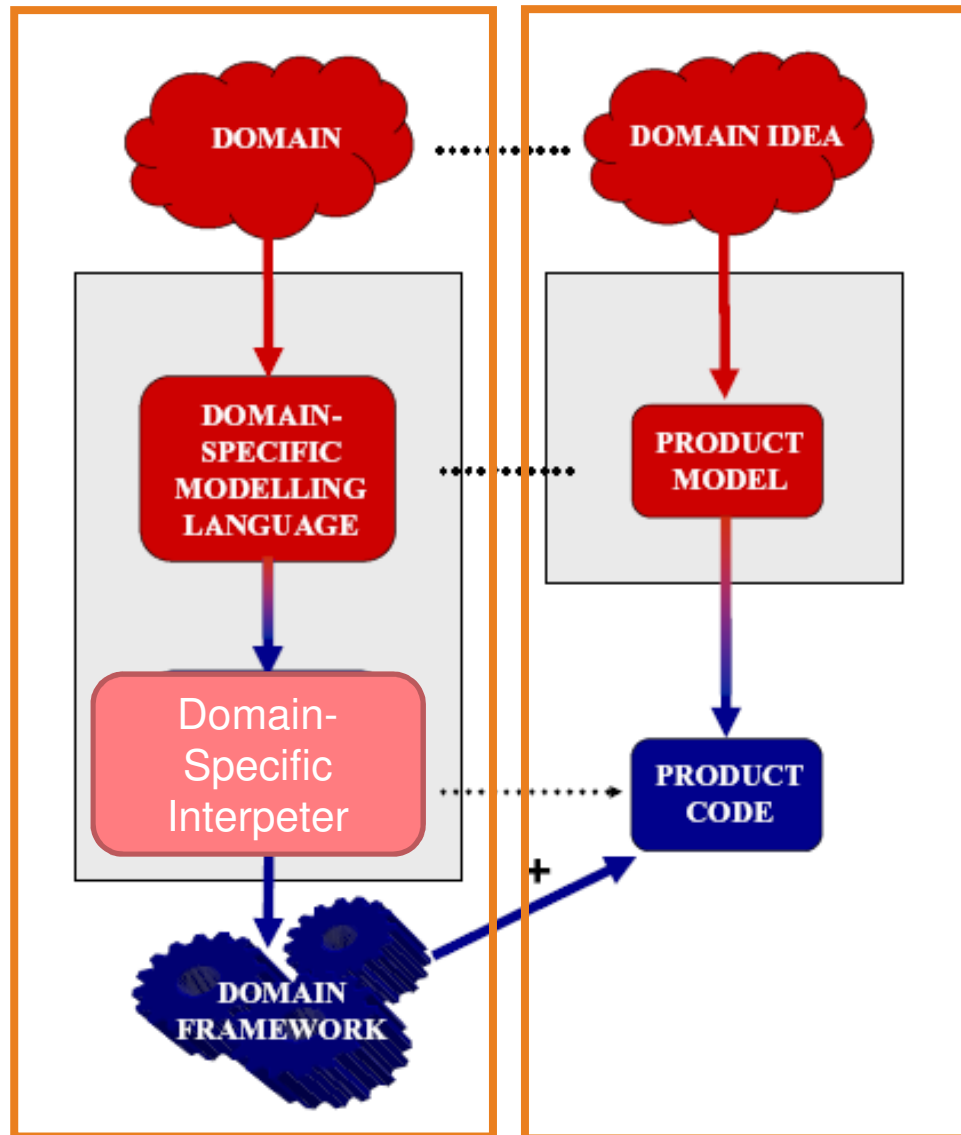
+ abstração e poder de expressão!

[Tolvanen 2003]

Modelagem Visual



Feito apenas uma vez, pelo projetista da linguagem



Feito várias vezes, pelos desenvolvedores

+ abstração e poder de expressão!

[Tolvanen 2003]

DSL textual - Exemplo



```
Define AnnotationShape Rectangle
    Width=1.5
    Height=0.3
    FillColor=khaki
    OutlineColor=brown
    Decorator Comment
        Position="Center"
    End Comment
End AnnotationShape
```

DSL textual - Dificuldades



- Criação de um parser
- Verificação semântica
- Escrever gerador de código ou interpretador
- Editor de texto, debugger,...

Embedded DSLs



- Criação de uma DSL dentro de outra linguagem
- Evita necessidade de criar parser e toda a infraestrutura de uma nova linguagem
- Algumas linguagens que facilitam este caminho:
 - Groovy, Haskell, logix, ...
 - XML.
 - Mas também é possível em Java, C# etc.

Embedded DSLs – XML



```
<?xml version="1.0" encoding="utf-8" ?>
<Shapes>
  <Shape name="AnnotationShape">
    <Kind>Rectangle</Kind>
    <Width>1.5</Width>
    <FillColor>Khaki</FillColor>
    <OutlineColor>Brown</OutlineColor>
    <Decorator name="Comment">
      <Position>Center</Position>
    </Decorator>
  </Shape>
</Shapes>
```

Linguagens com maior suporte a Embedded DSLs



- Poucas regras sintáticas
- Poucas palavras chave
- Pouca ou nenhuma distinção sintática entre linguagem, biblioteca e programa do usuário
- Sintaxe leve para funções anônimas
- combinadores
- monads

Embedded DSLs – C#



```
Shape AnnotationShape = new
    Shape(ShapeKind.Rectangle,
        1.5,
        0.3,
        Color.Khaki,
        Color.Brown
    );
```

```
Decorator Comment = new Decorator(Position.Center);
AnnotationShape.AddDecorator(Comment);
```

Projetando uma DSL



- Etapas sugeridas:
 - Análise
 - Implementação
 - Utilização

Projetando uma DSL :: Análise



- ❑ Objetivo: construir um entendimento do domínio
- ❑ Passos:
 1. Identificar o domínio do problema
 2. Obter o conhecimento relevante sobre esse domínio
 3. Juntar nesse conhecimento noções semânticas e operadores nessas noções - combinadores
 4. Projetar a DSL que descreve concisamente aplicações no domínio

Projetando uma DSL :: Implementação



1. Construir uma biblioteca (API) ou framework que implementa noções semânticas
2. Projetar e implementar um compilador (ou interpretador) que traduz (ou interpreta) programas na DSL em uma seqüência de chamadas a bibliotecas

Projetando uma DSL :: Utilização

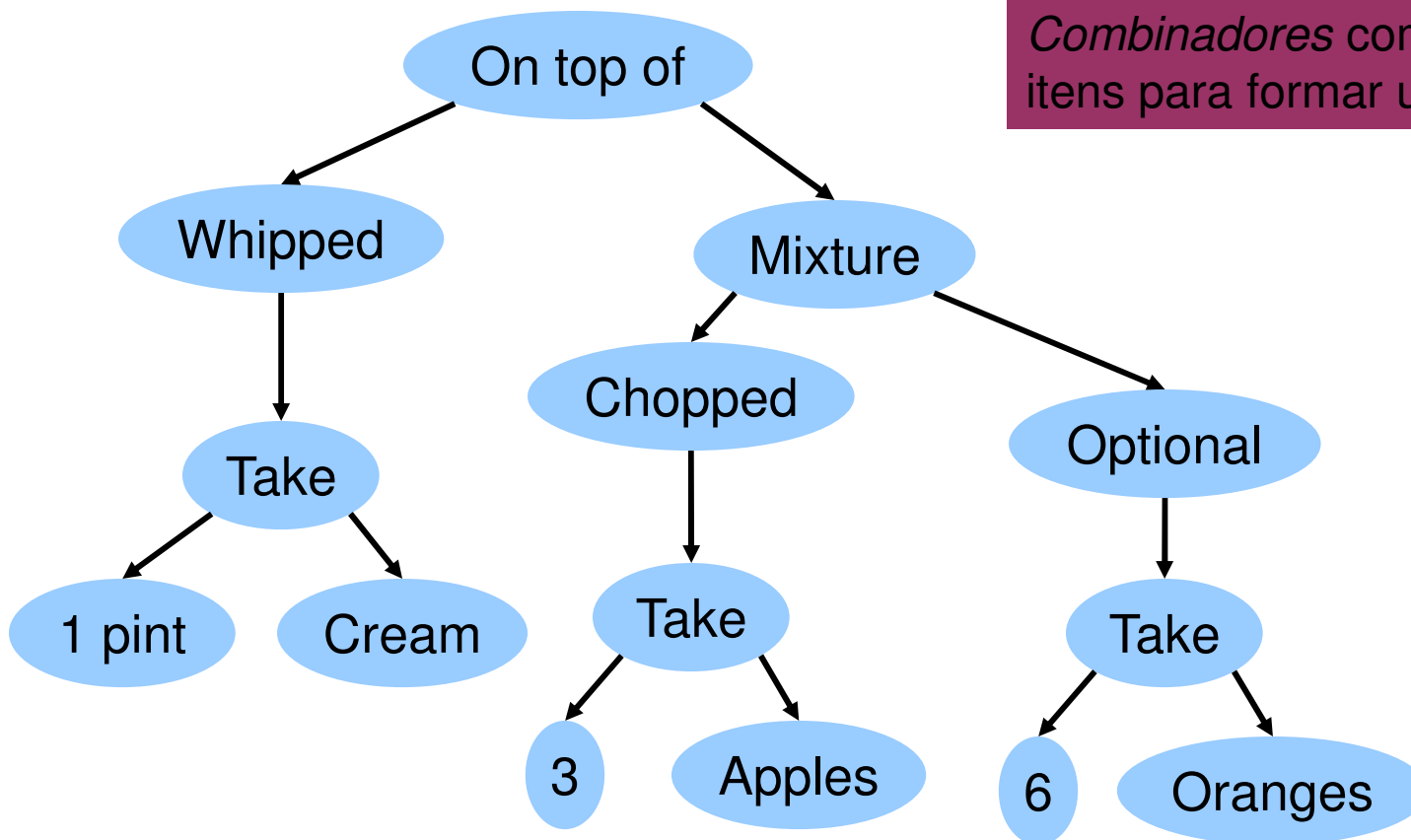


1. Escrever programas na DSL para as aplicações desejadas
2. (Interpretar ou) compilar e executar os programas
3. Validar a DSL

Exemplo – receitas de salada



Combinadores combinam itens para formar uma salada



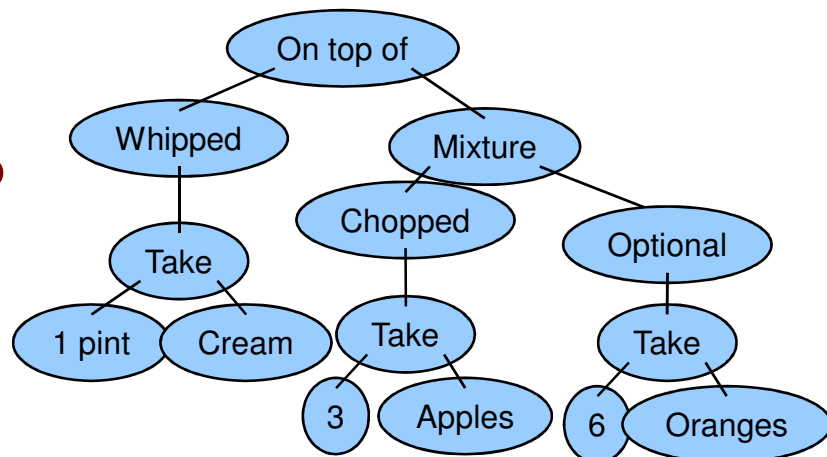
De [*Composing contracts: an adventure in financial engineering.*](#)

Simon Peyton Jones, Jean-Marc Eber, Julian Seward, ICFP 2000



Embedded DSLs - exemplo

```
salad      = topping `onTopOf` main_part
topping    = whipped (take pint cream)
main_part  = mixture apple_part orange_part
apple_part = chopped (take 3 apple)
orange_part = optional (take 6 oranges)
```



DSLs – suporte a interpretação



Solicitação: $S(P)$, conteúdo de açúcar de uma salada P

```
S (onTopOf p1 p2)      = S(p1) + S(p2)
S (whipped p)          = S(p)
S (take q i)           = q * S(i)
...etc...
```

Ao definir uma receita, podemos calcular seu conteúdo de açúcar sem trabalho extra.

Só se novos combinadores fores criados ou novos ingredientes é preciso estender S

Exemplo - contratos



“Receive £100 on 1 Jan 2010”

```
c1 :: Contract
c1 = zcb (date "1 Jan 2010") 100 Pounds
```

```
zcb :: Date -> Float -> Currency -> Contract
-- Zero coupon bond
```

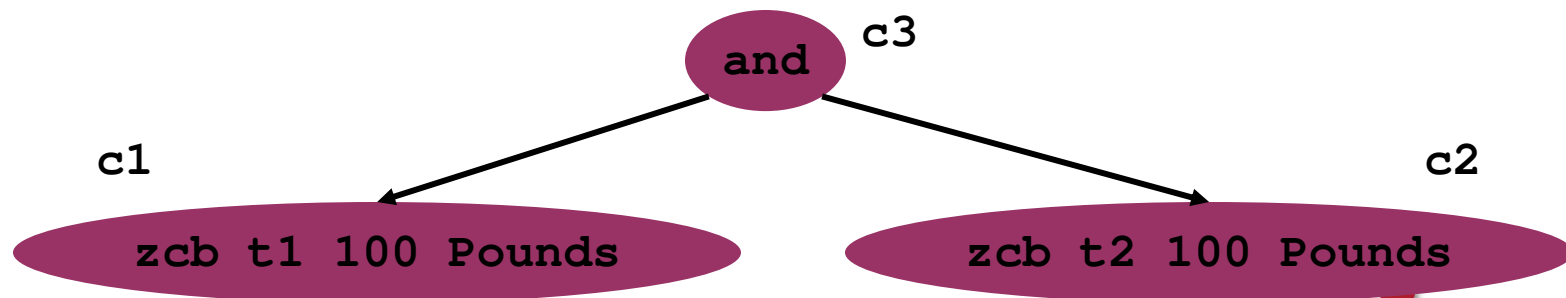


Combinando contratos

```
c1, c2, c3 :: Contract
c1 = zcb (date "1 Jan 2010") 100 Pounds
c2 = zcb (date "1 Jan 2011") 100 Pounds

c3 = c1 `and` c2
```

```
and :: Contract -> Contract -> Contract
-- Both c1 and c2
```



Resultados da DSL de contratos



- Pequeno conjunto de combinadores
- Biblioteca extensível pelo usuário
- Semântica composicional, facilita modelos de valoração
- Prêmio Produto de Software do Ano da Risk Magazine
- Empresa LexiFi, comercializa esta idéia

DSLs: riscos e oportunidades



- Vantagens:
 - Soluções no idioma e no nível de abstração do domínio do problema
 - Experts no domínio podem entender, modificar, validar e até fazer programas
 - Programas concisos, auto-documentados
 - Validação / otimização no nível do domínio
 - Conservam e reusam conhecimento do domínio

DSLs: riscos e oportunidades



- Desvantagens:
 - Custo de projeto, implementação e manutenção da DSL
 - Custo de treinamento de programadores da DSL
 - Dificuldade na definição do escopo da DSL
 - Dificuldade em balancear construtores de linguagens de programação com especificidades do domínio
 - Possível perda de performance

Reflexão



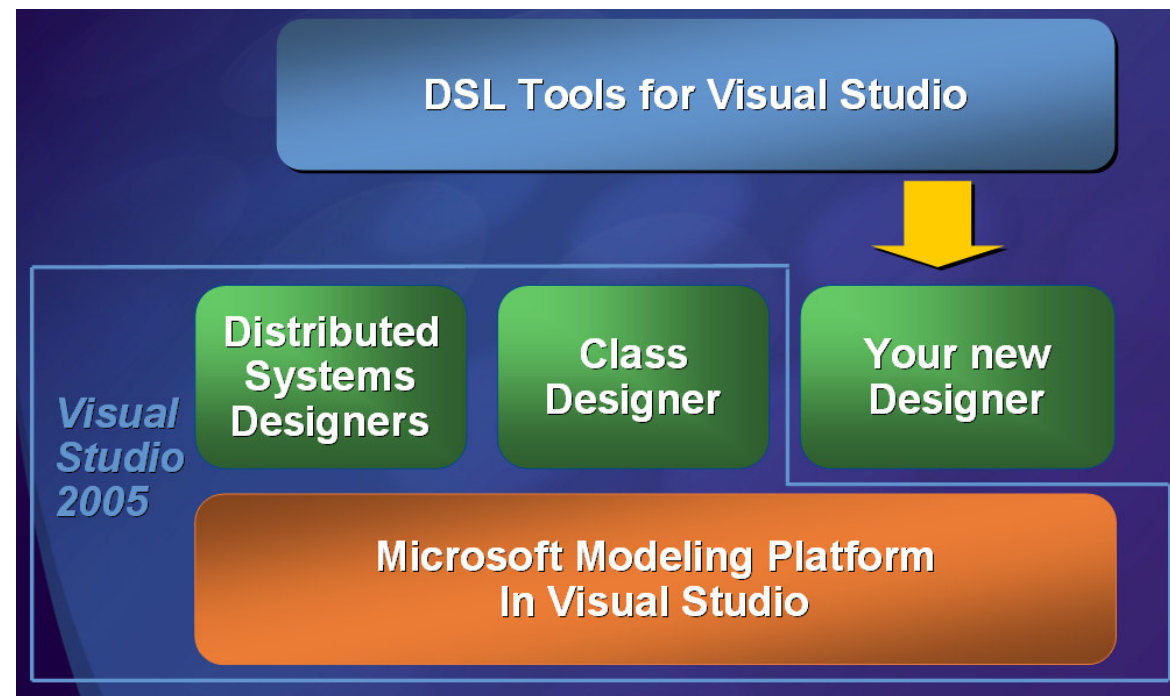
- DSLs (principalmente as visuais) são poderosas e ao mesmo tempo custosas... como melhorar isso?

**Através de programas de suporte à criação de DSLs:
“language workbenches”**

VS Team System DSL Tools



- Plug in para a IDE Visual Studio .NET
- Suporte à criação de **DSLs visuais**
 - Captura dos conceitos do domínio
 - Definição de um editor gráfico
 - Criação de transformadores (ex.: gerador de código)



Exemplo: DSL para Jogos – SharpLudus



The screenshot displays the SharpLudus IDE interface. The main workspace shows a game design workflow starting with an 'Intro screen' (a menu with options like 'START NEW GAME', 'HIGH SCORE BOARD', 'TELL-A-FRIEND THIS GAME', and 'QUIT GAME') that transitions to 'Room 1' and then to 'Room 2'. The rooms are detailed 2D environments with furniture and objects. The 'Toolbox' on the left lists components like 'Pointer', 'InfoDisplay', 'Transition', and 'Room'. The 'SLGML Explorer' on the right shows a tree view of the game's components. The 'Properties' panel for 'MyGame SharpLudusGame' lists various settings. The 'Error List' at the bottom shows two errors.

Error List

	Description	File	Line	Column	Project
1	This game contains no main character	MyGame.slgml	0	1	SLGMLDebugging
2	A game over info display must be specified	MyGame.slgml	0	1	SLGMLDebugging

Properties

MyGame SharpLudusGame

Property	Value
AudioComponents	(none)
Entities	(none)
Events	(none)
FullScreen	False
Name	MyGame
Namespace	MyNamespace
Resolution	640; 480
Sprites	(none)

DSLs :: Conceitos Relacionados



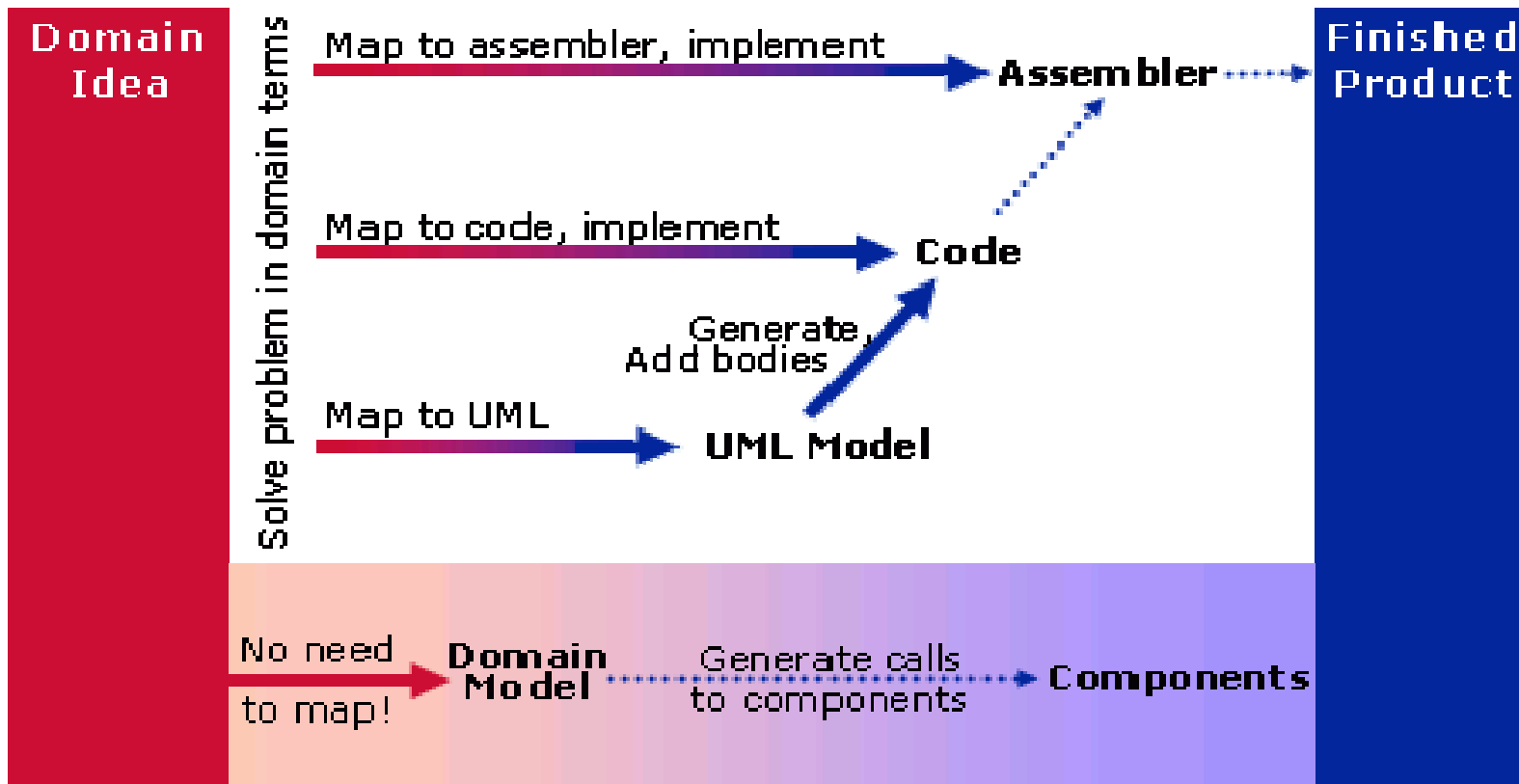
- Language-Oriented Programming
 - Estilo de desenvolvimento baseado na idéia de construir software através de DSLs
 - Proposto pelo fundador da JetBrains
- Language Workbench
 - Conjunto de ferramentas para viabilizar a language-oriented programming
 - Martin Fowler

DSLs :: Conceitos Relacionados



- Software Factories
 - Iniciativa da Microsoft para suporte a DSLs, da qual faz parte o DSL Tools para o Visual Studio.
- Eclipse Modeling Framework
 - <http://www.eclipse.org/modeling/emf/>
- Intentional Software
 - www.intentionalsoftware.com
 - Ambiente de desenvolvimento em que toda a programação é de domínio específico

E UML, MDA, MDD?



[Tolvanen 2003]

Conclusões



- DSLs podem aumentar significativamente a produtividade em projetos de um mesmo domínio (e.g. 5x, 10x)
- Mas é preciso avaliar bem o retorno do investimento em criar uma DSL (N projetos em um mesmo domínio)

A Good Programmer does Language Design

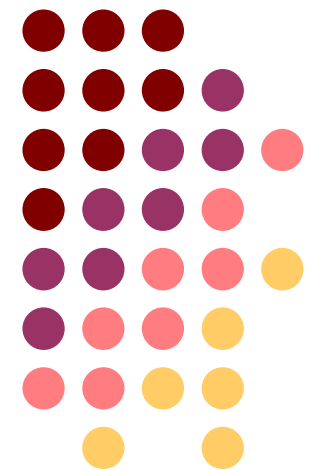


- “[...] a good programmer in these times does not just write programs. [...] a good programmer does language design, though not from scratch, but building on the frame of a base language.”
— Guy Steele Jr., Growing a language.
OOPSLA, 1998.

Perguntas?

André Santos – alms@cin.ufpe.br
André Furtado – awbf@cin.ufpe.br

CIn-UFPE





Referências

- **Domain-Specific Development with Visual Studio DSL Tools**, by Steve Cook, Gareth Jones, Stuart Kent, Alan Cameron Wills, Addison-Wesley, 2007.
- *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*, by Jack Greenfield and Keith Short, with Steve Cook and Stuart Kent, Wiley, 2004.



Referências

- VS 2005 Team System Domain-Specific Language (DSL) Tools
msdn.microsoft.com/vstudio/DSLTools/
- Projeto SharpLudus -
<http://www.cin.ufpe.br/~sharpludus>



Referências

- Language-Oriented Programming: The Next Programming Paradigm, Sergey Dimitriev, co-founder and CEO of JetBrains,
<http://www.onboard.jetbrains.com/is1/articles/04/10/lop/>
- Language Workbenches: The Killer-App for Domain Specific Languages?
<http://www.martinfowler.com/articles/languageWorkbench.html>