UFRPE
Universidade Federal Rural de Pernambuco

Paradigmas de Programação

Critérios de Avaliação das Linguagens

Kellyton Brito
kellyton.brito@gmail.com
06 e 10/10/2017

Muitas Linguagens e Paradigmas

- Funcional
  - LISP, ML, Miranda, Haskel …
- Lógico
  - Prolog, Godel …
- Imperativo
  - Fortran, Cobol, Natural, PL/I, Pascal, C …
- Orientado a Objetos
  - Simula, Smalltalk, C++, C#, Java …
- Orientado a Aspectos
  - AspectJ …

Dado um novo projeto de desenvolvimento…

Qual Linguagem escolher?

Qual paradigma/linguagem/framework?

- Sempre levando em consideração as necessidades do usuário/sistema!

- Análise do domínio da aplicação
  - Escolha do paradigma adequado

- Escolha da linguagem adequada
  - Critérios de avaliação das linguagens

Critérios de avaliação de linguagens

Se você fosse escolhido para desenvolver o

…

quais critérios você usaria para escolher a linguagem de programação a utilizar?

Critérios de Avaliação das linguagens

- Úteis para verificar o ***impacto*** da linguagem no processo de desenvolvimento

- Principais categorias de critérios:
  - Readability – Leitura
  - Writability – Escrita
  - Reliability – Segurança/Confiança
  - Cost – Custo
- Cada categoria tem subfatores interligados entre si

	Categoria		
Critério	Leitura	Escrita	Confiança
Simplicidade/Ortogonalidade	X	X	X
Estruturas de controle	X	X	X
Tipos de dados e estruturas	X	X	X
Sintaxe	X	X	X
Suporte a abstração		X	X
Expressividade		X	X
Checagem de tipos			X
Tratamento de exceções			X
"Aliasing"			X

	Categoria		
Característica	Leitura	Escrita	Confiança
Simplicidade/Ortogonalidade	X	X	X
Estruturas de controle	X	X	X
Tipos de dados e estruturas	X	X	X
Sintaxe	X	X	X
Suporte a abstração		X	X
Expressividade		X	X
Checagem de tipos			X
Tratamento de exceções			X
"Aliasing"			X

Readability – Simplicidade Geral

1. **Linguagens com muitas instruções básicas são mais difíceis de aprender e entender**
   - Desenvolvedores aprendem e usam o básico, e ignoram o resto

Readability – Simplicidade Geral

1. **Linguagens com muitas instruções básicas são mais difíceis de aprender e entender**
   - Desenvolvedores aprendem e usam o básico, e ignoram o resto

2. Multiplicidade: Muitas maneiras de fazer a mesma coisa

```
count = count + 1
count += 1
count++
++count
```

Readability – Simplicidade Geral

3. Sobrecarga de operador
   - Principalmente quando criados pelo usuário
   - Sobrecarga de + para adição de inteiros e Strings é previsível
   - Ex. ruim: Sobrecarga de + para adição de arrays não é previsível

Readability - Ortogonalidade

- Como é possível a ***combinação de tipos primitivos e estruturas***

  Todas as combinações são possíveis ou existem exceções?

- Exemplo:
  - Suponha uma linguagem com tipos primitivos *inteiro* e *char* e um tipo *array*.
    - Alta ortogonalidade: Todas as combinações: array de inteiro, array de char e array de array

Readability - Ortogonalidade

- Problemas com **baixa ortogonalidade**:

Cria-se **exceções** que o desenvolvedor tem que decorar para desenvolver corretamente.

	Categoria		
Característica	Leitura	Escrita	Confiança
Simplicidade/Ortogonalidade	X	X	X
Estruturas de controle	X	X	X
Tipos de dados e estruturas	X	X	X
Sintaxe	X	X	X
Suporte a abstração		X	X
Expressividade		X	X
Checagem de tipos			X
Tratamento de exceções			X
"Aliasing"			X

Readability – Estruturas de Controle

- Extinção do uso do ***goto***
  - A leitura completa de um programa do início ao fim é mais adequada do que pulos causados pelo ***goto***

- A estruturação e implementação de estruturas de controle facilitam leitura e entendimento

  if-then-else (break, continue), while, do-while, switch

	Categoria		
Característica	Leitura	Escrita	Confiança
Simplicidade/Ortogonalidade	X	X	X
Estruturas de controle	X	X	X
Tipos de dados e estruturas	X	X	X
Sintaxe	X	X	X
Suporte a abstração		X	X
Expressividade		X	X
Checagem de tipos			X
Tratamento de exceções			X
"Aliasing"			X

Readability – Tipos e Estruturas de Dados

- A existência de tipos de dados adequados é importante
  - Ausência do **booleano** em algumas linguagens
  - Ausência de strings, representadas por arrays de char
  - Representação aritmética (ponto flutuante)
- Necessidade de mecanismos adicionais para manipulação de dados
- Tipos de dados compostos, como records e classes, facilitam a estruturação

	Categoria		
Característica	Leitura	Escrita	Confiança
Simplicidade/Ortogonalidade	X	X	X
Estruturas de controle	X	X	X
Tipos de dados e estruturas	X	X	X
Sintaxe	X	X	X
Suporte a abstração		X	X
Expressividade		X	X
Checagem de tipos			X
Tratamento de exceções			X
"Aliasing"			X

Readability - Sintaxe

- Compreende a ***forma*** dos elementos da linguagem
- Possibilidade de representação próxima à natural
  - Limitação de tamanho dos nomes
  - Case sensitivity
  - Palavras reservadas e especiais
  - Delimitadores de final de sentença
  - Quantidade de instruções por linha

	Categoria		
Característica	**Leitura**	**Escrita**	**Confiança**
Simplicidade	X	X	X
Ortogonalidade	X	X	X
Estruturas de controle	X	X	X
Tipos de dados e estruturas	X	X	X
Sintaxe	X	X	X

Quais os principais *tradeoff's* (perde/ganha) dos critérios de leitura?

Writability - Escrita

- ***Facilidade*** com que uma linguagem pode ser usada para a ***criação*** de um programa para determinado domínio

- Para escrita é necessária a ***re-leitura*** de trechos do programa

	Categoria		
Característica	**Leitura**	**Escrita**	**Confiança**
Simplicidade	X	X	X
Ortogonalidade	X	X	X
Estruturas de controle	X	X	X
Tipos de dados e estruturas	X	X	X
Sintaxe	X	X	X
Suporte a abstração		X	X
Expressividade		X	X
Checagem de tipos			X
Tratamento de exceções			X
"Aliasing"			X

Writability – Suporte a Abstração

- Habilidade de utilizar estruturas complexas ignorando seus detalhes de implementação

- Pode-se abstrair:
  - Processos: algoritmos, funções, procedures
  - Dados: estruturas de dados, classes

- Escrita é feita uma vez, e reutilizada diversas vezes

Writability - Expressividade

- Quando a linguagem tem maneiras sucintas e eficientes de expressar a computação

- Exemplos:
  - Geralmente *count*++ é mais conveniente que *count = count + 1;*
  - Geralmente utilizar FOR é mais conveniente do que WHILE em muitas situações

Critérios de Avaliação das linguagens

Característica	Leitura	Escrita	Confiança
Simplicidade	X	X	X
Sintaxe	X	X	X
Suporte a abstração		X	X
Expressividade		X	X
Checagem de tipos			X
Tratamento de exceções			X
"Aliasing"			X

Categoria

Quais os principais *tradeoff's* dos critérios de escrita?

	Categoria		
Característica	Leitura	Escrita	Confiança
Simplicidade/Ortogonalidade	X	X	X
Estruturas de controle	X	X	X
Tipos de dados e estruturas	X	X	X
Sintaxe	X	X	X
Suporte a abstração		X	X
Expressividade		X	X
Checagem de tipos			X
Tratamento de exceções			X
"Aliasing"			X

Reliability - Confiabilidade

Um programa é dito confiável se <u>executa</u> sua especificação sob <u>todas</u> as condições de <u>contexto</u> possíveis.

Reliability – Checagem de Tipos

- Verificação e teste de erros de tipos de um programa
  - Detecta erros de atribuição, passagem de parâmetros, etc

- Tempo de compilação x tempo de execução
  - Tempo de compilação: Mais fácil de corrigir
    - Ex.: C/C++, Java
  - Tempo de execução: Mais liberdade ao desenvolvedor
    - Ex.: Python permite *binding* em tempo de execução

Reliability – Tratamento de Exceções

- Interceptação de erros em tempo de execução, tratá-los e continuar a execução

- Evita o mau funcionamento devido a fatores externos:
  - Conexão com banco, entrada/saída inconsistentes pelo usuário, permissões do sistema operacional…

Reliability – *Aliasing (apelido)*

- Mais de uma maneira de referenciar o mesmo dado (ou posição de memória)

- Exemplo: dois ponteiros apontando para o mesmo dado
  - Sempre que altera uma variável, altera o valor da outra

- Prática perigosa e danosa à confiabilidade do sistema

Critérios de Avaliação das linguagens

Característica	Leitura	Escrita	Confiança
Simplicidade	X	X	X
Sintaxe	X	X	X
Suporte a abstração		X	X
Expressividade		X	X
Checagem de tipos			X
Tratamento de exceções			X
"Aliasing"			X

Categoria

Quais os principais *tradeoff's* dos critérios de confiança?

Critérios de Avaliação das linguagens

- Principais categorias de critérios:
  - Readability – Leitura
  - Writability – Escrita
  - Reliability – Confiabilidade
  - ***Cost – Custo***

CUSTO

O custo é o conjunto complexo de **fatores** que influenciam na escolha da linguagem e ambiente de desenvolvimento

Tempo e esforço *para a construção, execução e manutenção dos sistemas*

Custos associados às linguagens

- Custo de escrita
  - Custo para desenvolver/escrever o software
  - Dependente das capacidades de leitura, escrita, e da <u>adequação ao domínio da aplicação</u>

- Custo de manutenção
  - Correção de bugs, inclusão de funcionalidades e migração para novas plataformas
  - Função principalmente das capacidades de **leitura** e das **ferramentas de apoio** disponíveis

Custos associados às linguagens

- Custo da confiabilidade
  - Afeta a manutenção do sistema
  - Afeta o negócio do cliente como um todo
  - **Pode *inviabilizar* a solução!**

- Custo do ambiente de desenvolvimento
  - Custo do software e do hardware necessário ao desenvolvedor
  - Existência e custo de ferramentas integradas, personalizáveis, evolutivas

Custos associados às linguagens

- Custo de Treinamento
  - Dependente da simplicidade, ortogonalidade e experiência da equipe
  - Influencia em todas as outras características

- Portabilidade
  - Aplicações poderem ser usadas em diferentes plataformas (Windows vs Linux, Android x iOS)

Custos associados às linguagens

- Generalidade
  - Uso em diversos tipos de aplicações
- Documentação
  - Documentação útil e completa
- Apoio da “comunidade”
  - Existência de forums e listas de discussão ativos, stackoverflow, etc

Custos associados às linguagens

- Custo de compilação
  - Esforço para geração de código de máquina
  - Interfere no desenvolvimento e testes

- Custo de execução
  - Esforço para execução da aplicação
  - Interfere no ambiente de execução
  - Envolve Compilação x Interpretação, Máquinas virtuais, checagem de tipos…

Design das linguagens

- O método de compilação/execução das linguagens afeta muitas características
  - Confiabiliadde
  - Custos de compilação
  - Custos de execução
  - Ambiente de desenvolvimento
  - Portabilidade

- Métodos de compilação/execução
  - Compilação
  - Interpretação
  - Híbrido

Métodos de Implementação

- **Compilação:**
  - Programas são **convertidos** diretamente para **linguagem de máquina**, que podem ser executados diretamente pelo sistema operacional
  - A compilação do programa todo é **executada de uma só vez**, e antes de sua execução
  - Programas rodam **mais rápido** na plataforma
  - Podem ocorrer **otimizações** de tempo e/ou espaço

Métodos de Implementação

- **Interpretação**:
  - O código fonte é lido e convertido para linguagem de máquina em "**tempo de execução**" da aplicação
  - É necessário um **interpretador** para, enquanto o programa é rodado, ler o código fonte e convertê-lo para código de máquina
  - Se existirem interpretadores para as plataformas alvo, uma aplicação **não precisa ser reescrita/recompilada para diferentes plataformas**
  - Aplicações executam mais **lentamente**

Métodos de Implementação

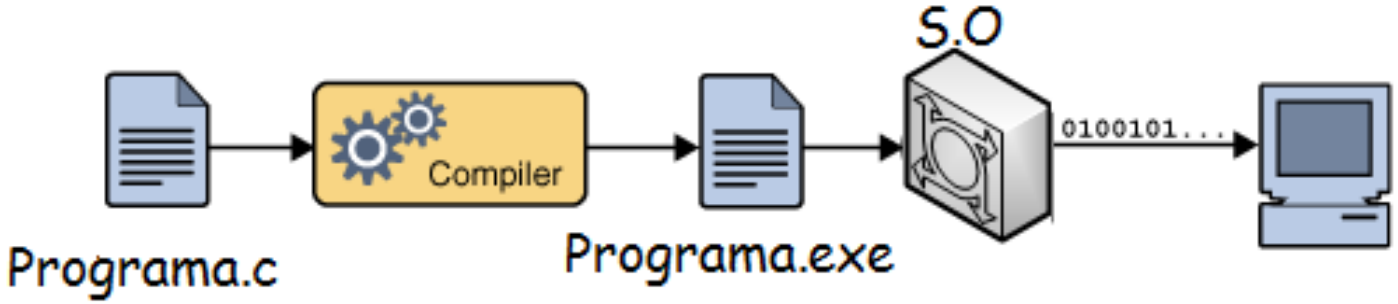
- **Híbrido**
  - Ocorre uma pré-compilação, com otimização
  - O código pré-compilado é lido e convertido para código de máquina em tempo de execução
  - Performance razoável
  - Mantém a portabilidade

Comparativo

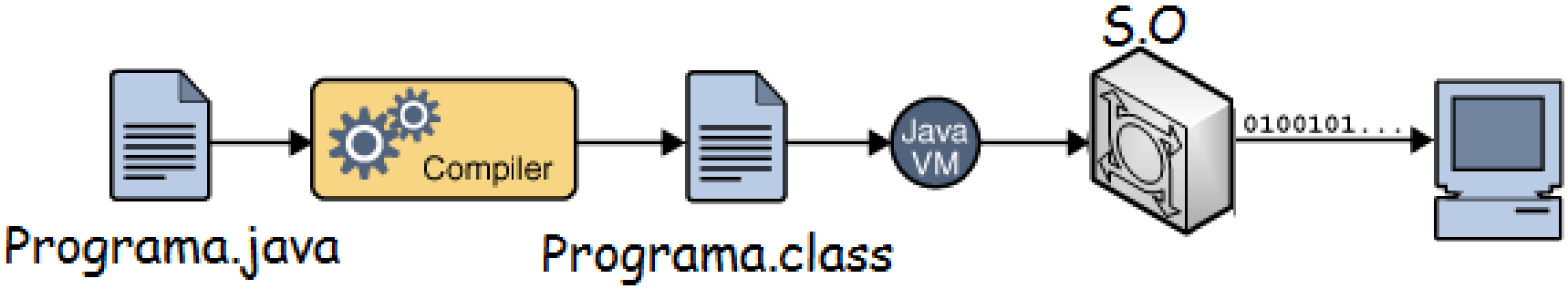
- Compilação:
  - Mais rápido, mais otimizado, mais poder ao desenvolvedor, protege o código fonte
  - Custo de hardware e software maior pelo desenvolvedor
- Interpretação
  - Mais confiável, mais portável, disponibiliza o código fonte
  - Custo de hardware e software maior pelo usuário
- Híbrido
  - Mais rápido que o interpretado, mais lento que o compilado
  - Protege o código fonte
  - Portável
  - Custo de hardware e software tanto pelo desenvolvedor quanto pelo usuário

Métodos de Implementação

- Por que dizem que C/C++ é mais rápido do que java?
- C/C++ é Compilado



- Java é Híbrido



Conflito nos Critérios

- Critérios conflitantes no design das linguagens
  - Alguns óbvios, outros mais detalhados
  - Confiabilidade x Custo de Execução
    - Problema dos arrays em C, inexistente em Java
  - Tipos de dados x Custo de Execução
    - Criação de tipos abstratos é mais custoso
  - Simplicidade x Expressividade
  - Ortogonalidade x Confiabilidade
  - Aliasing x Expressividade
  - …

Próxima Atividade

Se você fosse escolhido para desenvolver o

…

quais critérios você usaria para escolher a linguagem de programação a utilizar?