Universidade do Vale do Itajaí

Disciplina: Paradigmas de Programação

Professora: Fernanda Cunha

Aluna: Mariana Ferreira

RESUMO:

CRITÉRIOS DE AVALIAÇÃO DE LINGUAGENS

A sessão 1.3 do capítulo 1 do livro CONCEITOS DE LINGUAGEM DE PROGRAMAÇÃO de Robert Sebesta, trata dos principais critérios de avaliação das linguagens de programação. O autor destaca a importância desses critérios no impacto do desenvolvimento e da manutenção dos programas. Além disso, cita as principais características que os influenciam, que são a simplicidade, ortogonalidade, tipos de dados, projeto de sintaxe, suporte para abstração, expressividade, verificação de tipos, tratamento de exceções e apelidos restritos. O capítulo está dividido em quatro partes em que o autor aborda quatro desses principais critérios e explicada sobre as características citadas anteriormente.

- **Legibilidade**

A legibilidade é relacionada com a facilidade com que os programas podem ser lidos e entendidos. Antes de 1970 não era pensado na manutenção de código, os programas eram feitos apenas para serem eficazes e entendidos pela máquina. Entretanto, depois disso o conceito de ciclo de vida de software foi desenvolvido e foi entendido que a manutenção é uma parte muito importante no desenvolvimento de software. Surge então a necessidade de que os códigos sejam legíveis e fáceis de entender.

Principais características que contribuem para que uma linguagem seja legível:

1. Simplicidade Geral

A simplicidade é um fator importante para legibilidade de uma linguagem. Se uma linguagem tem um grande número de componentes básicos ela é mais difícil de ser aprendida do que uma que tem poucos desses componentes. O programador não consegue aprender todos esses componentes e acaba aprendendo um subconjunto, diferentes programadores que conhecem diferentes subconjuntos desses componentes vão ter mais dificuldade de entender os códigos uns dos outros.

Já a multiplicidade de recursos é um fator que pode prejudicar a legibilidade de um programa. É quando se tem diferentes formas de fazer uma mesma operação. Em Java e C++ por exemplo, existem quatro diferentes formas de incrementar ou decrementar uma variável.

Outro fator que pode dificultar a legibilidade de um programa é a sobrecarga de operadores, que é quando um mesmo operador pode fazer diferentes coisas dependendo do contexto em que é usado. O operador +, por exemplo, pode ser usado tanto para soma de valores inteiros quando valores flutuantes, o que é bastante intuitivo, por se tratar de valores

numéricos. Porém, quando esse operador começa a ser usado também para fazer outros tipos de operações em outras estruturas de dados a legibilidade do programa pode se tornar um pouco mais difícil.

Uma linguagem simples que facilite a legibilidade do código é muito importante, porém se a linguagem for simples demais, a estruturação do código pode acabar se tornando muito complexa, o que também prejudica a legibilidade.

2. Ortogonalidade

Ortogonalidade em linguagens de programação significa poder combinar diferentes tipos de dados. Se houver exceções do que pode ser combinado muitas das possibilidades de estruturas possíveis são eliminadas, o que faz com que tenhamos uma falta de ortogonalidade.

A ortogonalidade também está fortemente relacionada a simplicidade, pois quanto menores forem as exceções mais fácil se torna o aprendizado e o entendimento da linguagem.

Existem alguns exemplos de falta de ortogonalidade em C. Como, dos dois tipos de dados estruturados em C (vetores e structs) somente structs podem ser retornadas por funções. Parâmetros de funções são passados por valor, exceto vetores.

Porém muita ortogonalidade também pode dificultar a leitura, porque torna mais difícil encontrar erros já que todas as combinações são permitidas.

3. Tipos de dados

Um maior número de tipos e estruturas de dados auxiliam na legibilidade. Usando valores booleanos “True” e “False” ao invés de valores inteiros 0 e 1, a intenção do código fica muito mais clara e legível para outros programadores.

4. Projeto da sintaxe

A sintaxe de uma linguagem tem um papel fundamental na sua legibilidade.

Restrições de tamanho dos identificadores, por exemplo, dificultam que as variáveis tenham nomes significativos, o que prejudica o entendimento do que está sendo feito.

As palavras especiais também impactam na legibilidade do código. Quando se tem várias estruturas aninhadas, por exemplo, o uso de chaves, que é usado por muitas linguagens, dificulta saber qual é o fim de cada estrutura, o uso de palavras reservadas compostas como “end loop” e “end if” auxiliam a leitura nesse aspecto.

Se palavras especiais podem ou não ser usadas como nome de variáveis, também impacta a leitura, já que um código em que a mesma palavra pode ou não significar algo especial deixa a leitura mais confusa.

O princípio de forma e significado é violado quando duas formas parecidas possuem resultados diferentes dependendo do contexto em que estão inseridas.

- **Facilidade de escrita**

A facilidade de escrita e legibilidade compartilham muitas coisas em comum. A principais características de uma linguagem com facilidade de escrita são:

1. Simplicidade e ortogonalidade

Uma linguagem que tenha muitos elementos torna a escrita de código mais difícil, pois o programador muito provavelmente não vai conhecer todas as possibilidades da linguagem e pode acabar não usando as mais adequadas ou até mesmo usando de forma equivocada. Por isso, é importante que a linguagem tenha uma simplicidade e utilize da ortogonalidade para combinar as estruturas para que se construa um maior número de instruções.

2. Suporte à abstração

A linguagem ter um suporte a abstração é um fator muito importante na facilidade de escrita. Abstração é a capacidade de definir e usar estruturas complexas ignorando os detalhes que não são relevantes para o problema em questão, o que torna a escrita do código muito mais simples.

3. Expressividade

Um programa mais expressivo é aquele que tem a capacidade de fazer uma grande quantidade de processamento em um programa menor, reduzindo comandos ou especificando a computação de uma forma conveniente.

- **Confiabilidade**

Um programa é confiável quando ele se comporta de acordo com suas especificações em todas as condições. Alguns recursos que ajudam com a confiabilidade do programa:

1. Verificação de tipos

A confiabilidade na verificação de tipos é feita através de testes para ver se existem erros de tipos no programa por meio do compilador ou durante a execução.

A verificação em tempo de compilação é mais desejável porque quando mais cedo os erros forem detectados menos caro será para fazer os reparos.

2. Tratamento de exceções

A capacidade de detectar erros que iriam interromper o programa e em tempo de execução interceptar o erro e resolver ou mostrar uma mensagem de erro auxilia muito na confiabilidade de um programa.

3. Utilização de apelidos

Apelidos são permitidos quando é possível ter nomes distintos fazendo referência a mesma área de memória, o que é um recurso perigoso e diminiu a confiabilidade de um programa.

4. Legibilidade e facilidade de escrita

A clareza na leitura e a facilidade de escrita são cruciais para garantir a confiabilidade de um programa. Uma linguagem que não proporciona maneiras naturais de expressar algoritmos leva a abordagens menos intuitivas, aumentando a probabilidade de erros. Quanto mais simples for escrever um programa, maior a probabilidade de ele estar correto.

- **Custo**

Principais custos de um desenvolvimento de software:

1. Custo de treinamento de programadores, que envolve a simplicidade e a ortogonalidade. Linguagens mais poderosas geralmente trazem um custo maior nesse aspecto.
2. Custo para escrever programas na linguagem (capacidade de escrita). As linguagens de alto nível vieram para tentar diminuir o custo nesse sentido.
3. Custo para compilar programas na linguagem.
4. Custo para executar programas (influenciado pelo projeto da linguagem). Uma linguagem que requer muitas verificações vai ter uma execução mais lenta. Uma técnica usada para corrigir isso é chamada de otimização.
5. Custo do sistema de implementação da linguagem. Uma linguagem que pode ser implementada em plataformas mais baratas tem mais chances de se tornarem populares.
6. Custo da má confiabilidade. A falha de um software pode ser muito custosa, seja esse software um sistema crítico, como usinas nucleares ou equipamentos médicos, ou outros tipos de sistemas que não tem uma criticidade, mas ainda sim sua falha trará custos.
7. Custo de manutenção de programas. Esse custo inclui tanto as correções quanto as melhorias. Aqui, quando a linguagem não tem uma boa legibilidade, o custo aumenta muito.

Outros critérios ainda mencionados pelo autor são a portabilidade, a generalidade e a linguagem ser bem definida. Por fim, ele enfatiza a importância desses conceitos para a avaliação de linguagens de programação.