Uma Linguagem Multiparadigma para o Ensino Integrado de Engenharia de Software e Programação Funcional

Leonardo R. Lucena[1], **Daniel Lucena**[2]

[1]Diretoria Acadêmica de Informática
Instituto Federal do Rio Grande do Norte (IFRN)
Av. Salgado Filho, 1559 – 59015-000 – Natal – RN – Brazil

[2]Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte (UFRN)
Av. Salgado Filho, s/n. – 59078-970 – Natal – RN – Brazil

Abstract. *This article proposes a pedagogical approach for the integrated teaching of Software Engineering and Functional Programming through the Potigol language. Its functional features are highlighted, favoring quality attributes such as testability, clarity, and modularity. A teaching model is presented that articulates Software Engineering concepts with functional principles. The multiparadigm nature of the language, its pedagogical implications, and perspectives for future evaluations are also discussed.*

Resumo. *Este artigo propõe uma abordagem pedagógica para o ensino integrado de Engenharia de Software e Programação Funcional por meio da linguagem Potigol. Destacam-se seus recursos funcionais, que favorecem atributos de qualidade como testabilidade, clareza e modularidade. Apresenta-se um modelo de ensino que articula conceitos de Engenharia de Software com princípios funcionais. Discutem-se ainda a natureza multiparadigma da linguagem, suas implicações pedagógicas e perspectivas para futuras avaliações.*

1. Introdução

Ensinar programação a iniciantes é um desafio reconhecido [Castro et al. 2002]. Embora cursos introdutórios geralmente priorizem o paradigma imperativo, pesquisas indicam que paradigmas alternativos podem facilitar o aprendizado [Pérez and López 2007]. As diretrizes curriculares nacionais para Computação [BRASIL 2016] recomendam abordar também paradigmas como o funcional e o orientado a objetos antes do estudo detalhado das linguagens. Além disso, o ensino de hoje mostra que programar não é só especificar algoritmos, mas produzir artefatos de software com qualidade, tais como modularidade, clareza, testabilidade e facilidade de manutenção [Hughes 1990, Castro et al. 2002].

Neste contexto, apresentamos a linguagem Potigol como ferramenta didática para integrar o ensino de Engenharia de Software (ES) e Programação Funcional (PF). Potigol é uma linguagem educacional, de propósito geral, com sintaxe intuitiva em português, suporte a múltiplos paradigmas (funcional, imperativo e orientado a objetos) e tipagem estática com inferência de tipos [Potigol 2021]. Este trabalho consolida propostas anteriores [Lucena and Lucena 2016], adicionando novos elementos multiparadigma e fundamentando metodologicamente a abordagem.

2. Fundamentação Teórica

Integrar Engenharia de Software (ES) e Programação Funcional (PF) no ensino busca explorar sinergias entre qualidade de software e paradigmas de programação. Na ES, destacam-se atributos como testabilidade, modularidade, manutenibilidade e legibilidade [Khanfor and Yang 2017, Hughes 1990]. Já o paradigma funcional incentiva funções puras (sem efeitos colaterais) e valores imutáveis, gerando código mais determinístico e fácil de testar [Hu et al. 2015]. Além disso, a expressividade e abstrações matemáticas do estilo funcional tornam o código mais claro e favorecem a manutenção.

Pesquisas educacionais mostram que o paradigma funcional tem vantagens no ensino inicial. Castro et al. (2002) destacam que a PF se baseia em conceitos matemáticos familiares aos alunos (como funções, mapeamentos) e possui elevado poder de expressão, permitindo abordar problemas complexos desde o início. Além disso, por ser menos conhecido dos iniciantes, o estilo funcional nivela a turma, pois todos começam sem experiência prévia, aplicando raciocínios matemáticos de forma natural. No contexto de ES, a adoção de funções puras e imutabilidade pode reduzir a carga cognitiva na transformação de soluções em instruções de programação, reforçando o foco na modelagem formal dos problemas [Castro et al. 2002, Hu et al. 2015].

Trabalhos anteriores analisam estratégias para o ensino de paradigmas. Pérez & López (2007) criticam a ênfase no paradigma orientado a objetos como "primeiro paradigma" nos currículos e defendem o uso de linguagens multiparadigma para explorar diferentes estilos de programação. Castro et al. (2002) relatam a experiência de introduzir o paradigma funcional em cursos iniciais, destacando o foco em modelagem de problemas e abstração lógica. Em comum, esses estudos apontam que abordagens pedagógicas mais eficazes combinam a resolução de problemas reais com a experimentação em diferentes paradigmas, reforçando o raciocínio matemático (característico da PF) e práticas de desenvolvimento (típicas da ES).

Em síntese, a literatura indica que o ensino integrado de ES e PF beneficia iniciantes: princípios funcionais apoiam práticas de ES (como testes unitários e modularização), enquanto as noções de qualidade (clareza, modularidade) são fortalecidas pelo paradigma funcional.

3. A Linguagem Potigol

Potigol é uma linguagem de programação multiparadigma criada para cursos introdutórios de programação [Lucena and Lucena 2016]. Projetada para iniciantes, possui sintaxe intuitiva em português, poucas construções sintáticas (reutilizadas em diferentes contextos) e tipagem estática com inferência de tipos, combinando suporte aos paradigmas funcional, imperativo e orientado a objetos. A sintaxe e os recursos de Potigol foram refinados iterativamente com base em feedback de sala de aula.

Embora seja multiparadigma, Potigol prioriza e incentiva o uso do paradigma funcional. A Listagem 1 exemplifica algumas das principais características funcionais da linguagem, como funções puras (linha 3) e de alta ordem (linhas 8 e 11), imutabilidade por padrão (linha 2), compreensão de listas (linha 16) e casamento de padrões (linhas 22-28).

Potigol permite definir funções por expressões lambda, atribuí-las a variáveis (linha 4), passá-las como parâmetro (linha 9) e retorná-las de outras funções (linha 11),

facilitando a composição e a reutilização de código. O casamento de padrões possibilita definições concisas e legíveis de funções recursivas e desestruturação de listas, tuplas e objetos. A sintaxe em português torna o código mais compreensível no contexto pedagógico brasileiro, reduzindo a barreira da língua.

Listagem 1. Características Funcionais de Potigol

```
# Variável imutável e função simples
a = 10
soma(x, y: Inteiro) = x + y
prod = (x, y: Inteiro) => x * y                # Expressão lambda

# Funções de alta-ordem
tipo FunBin = (Inteiro, Inteiro) => Inteiro
hof(f: FunBin, x, y: Inteiro) = f(x, y)
escreva hof(prod, 10, 20)                      # Saida: 200

soma1(x: Inteiro) = (y: Inteiro) => x + y      # retorna uma função
sucessor = soma1(1)                            # define sucessor
escreva sucessor(3)                            # Saída: 4

# Manipulacao de Listas
numeros = para i de 1 ate 6 gere i fim         # [1, 2, 3, 4, 5, 6]
pares = numeros.selecione(n => n mod 2 == 0)   # filtro de pares
soma  = numeros.injete(0)((a, b) => a + b)     # soma/fold(injete)
dobro = numeros.mapeie(n => 2 * n)             # mapeamento

# Casamento de Padrões com listas
quicksort(nums: Lista[Inteiro]): Lista[Inteiro] = escolha nums
  caso []           => []                      # Lista vazia
  caso pivot::resto =>                         # Lista não vazia
    menores = resto.selecione(_ < pivot)
    maiores = resto.selecione(_ > pivot)
    quicksort(menores) + pivot :: quicksort(maiores)
fim
```

4. O Modelo de Ensino Integrado (ES + PF)

Propomos um modelo pedagógico que utiliza Potigol para integrar conceitos-chave de Engenharia de Software (ES) e programação funcional (PF). Os tópicos de ES considerados incluem testes automatizados, modularidade, organização e manutenção do código, além de clareza e legibilidade. Esses conceitos são trabalhados em conjunto com princípios de PF, como funções puras, imutabilidade, funções de alta ordem, recursão e casamento de padrões. A seguir, destacamos os principais elementos do modelo:

- **Testes e funções puras**: Funções puras, por serem determinísticas, são naturalmente adequadas para testes unitários. Recomenda-se introduzir testes simples logo após a definição das funções, estimulando o hábito de validação incremental.
- **Modularidade e funções de alta ordem**: O modelo incentiva dividir programas em funções e módulos coesos. As funções são cidadãos de primeira classe, permitindo passá-las como parâmetros e construir algoritmos genéricos (por exemplo, *map*, *filter*), reforçando a reutilização de código.
- **Clareza com imutabilidade e expressividade**: A imutabilidade evita efeitos colaterais que dificultam a lógica do programa. Estruturas como recursão e funções

de alta ordem substituem o uso de variáveis mutáveis. Além disso, o casamento de padrões torna o código mais direto e legível ao eliminar cadeias complexas de *if/else*. O estilo funcional resulta em código mais fácil de entender e manter.

- **Desenvolvimento e manutenção**: As práticas de ES, como dividir o sistema em módulos, manter padrões de codificação e refatorar funções para maior generalidade, são plenamente integráveis. A clareza e previsibilidade das funções puras facilitam o trabalho colaborativo e reduzem o risco de erros.
- **Projetos em equipe**: Em atividades práticas, os alunos podem desenvolver projetos completos, aplicando conceitos de ES (especificação, design, teste, integração) e PF (pureza, modularidade). Por exemplo, ao implementar uma agenda de contatos, grupos podem definir casos de teste, criar módulos e discutir a separação lógica entre componentes.

Assim, o modelo fortalece simultaneamente: (1) práticas de ES, como desenvolvimento orientado a testes (TDD), revisão de código e modularização; e (2) princípios de PF, como minimização de efeitos colaterais, uso de estruturas imutáveis e decomposição funcional. Conforme destacado em [Khanfor and Yang 2017, Hughes 1990], uma abordagem funcional moderna melhora o desenvolvimento de software com código mais limpo, maior escalabilidade e melhor manutenção, alinhando-se aos objetivos de qualidade de cursos de ES.

5. Aspecto multiparadigma de Potigol

Um dos diferenciais do Potigol é sua natureza multiparadigma: embora enfatize o paradigma funcional, também suporta construções imperativas e recursos orientados a objetos (OO). No estilo imperativo, comandos são usados principalmente para entrada/saída ou sequenciamento simples, como em `escreva "Digite um número:"`, o que facilita a transição para alunos acostumados a linguagens procedurais. No entanto, o modelo pedagógico destaca que essas construções podem ser substituídas por funções puras, promovendo melhores propriedades de clareza, testabilidade e manutenção.

Listagem 2. Classe em Potigol

```
1 tipo Retangulo
2   base, altura: Real
3   area() = base * altura
4 fim
5 retangulo = Retangulo(5.0, 4.0)
6 escreva "Area: {retangulo.area}"     # Saida "Area: 20.0"
```

No paradigma OO, Potigol permite definir tipos de dados e funções associadas (métodos) de forma simples. Na Listagem 2 definimos um tipo `Retangulo` (linhas 1-4) com dois atributos (`base` e `altura`) e um método `area`. Em seguida (linha 5), instanciamos um retângulo e exibimos a sua área (linha 6).

Esse recurso pode ser explorado para discutir modelagem de dados ou organização de código, especialmente para alunos já familiarizados com OO. Por exemplo, ao implementar um cadastro de clientes, o uso de classes pode ajudar a agrupar atributos e comportamentos, sem necessidade de explorar herança ou encapsulamento em um primeiro momento. A orientação pedagógica é clara: priorizar funções puras para a lógica principal, usar comandos imperativos apenas para interação com o usuário e introduzir OO quando for didaticamente relevante.

Essa flexibilidade permite realizar comparações diretas entre paradigmas em sala de aula. Por exemplo, um mesmo problema pode ser resolvido com recursão (FP) ou com laços *enquanto* (imperativo), estimulando a reflexão sobre clareza, testabilidade e adequação de cada abordagem. Conforme destacam Pérez & López (2007), o uso de uma linguagem multiparadigma no ensino permite explorar diferentes estilos de forma integrada e coesa.

6. Discussão e perspectivas futuras

A proposta apresentada neste artigo combina fundamentos teóricos e aplicações práticas, mas sua efetividade pedagógica ainda precisa ser melhor avaliada em sala de aula. Atualmente, já existe uma documentação básica da linguagem e mais de 800 exercícios resolvidos na plataforma Beecrowd [Potigol 2025], o que valida seu uso em atividades práticas e no ensino de programação e algoritmos.

Planeja-se realizar estudos de caso em disciplinas de programação e engenharia de software, coletando dados sobre a aprendizagem de conceitos funcionais e de ES. Espera-se que Potigol, com sua sintaxe amigável e características funcionais, motive os alunos e introduza maior rigor conceitual. Entre os desafios futuros, destacam-se a criação de materiais didáticos e o desenvolvimento de mais ferramentas de apoio.

Além disso, sugere-se investigar como o paradigma funcional impacta a compreensão de conceitos clássicos de ES, como manutenção de software legado ou uso de métricas de qualidade. Acredita-se que estudos futuros possam consolidar evidências de que o ensino integrado de ES e PF contribui para formar alunos com visão mais ampla e competências técnicas sólidas. Outro ponto de discussão é a generalização do modelo de ensino aqui proposto. Embora tenhamos focado em Potigol, os princípios delineados (começar com PF e gradualmente introduzir ES e outros paradigmas) podem ser adaptados para outras linguagens funcionais ou multiparadigmas.

7. Considerações Finais

Este artigo apresentou um arcabouço teórico e metodológico para o ensino integrado de Engenharia de Software (ES) e Programação Funcional (PF). Foram analisados fundamentos educacionais sobre o ensino de programação e descrito como as características de Potigol (funções puras, imutabilidade, multiparadigma, entre outras) podem apoiar o aprendizado conjunto desses temas. Propomos um modelo de aula que integra práticas de ES (testes, modularidade, código limpo) com princípios de PF, ilustrado com exemplos concretos. Também destacamos o uso didático dos estilos imperativo e orientado a objetos, explicando quando seu uso é vantajoso no processo de ensino-aprendizagem.

Conclui-se que Potigol é uma linguagem promissora como ferramenta didática em cursos de computação, unindo simplicidade sintática, fundamentação conceitual e flexibilidade de paradigmas. Seus recursos permitem tratar temas de ES de maneira mais formal, por exemplo, facilitando a verificação de correção de programas por meio de funções puras. No futuro, pretendemos avaliar empiricamente o modelo proposto e refiná-lo, contribuindo para a literatura sobre ensino de Programação Funcional e Engenharia de Software.

Referências

BRASIL (2016). Resolução cne/ces nº 5, de 16 de novembro de 2016: Diretrizes curriculares nacionais para os cursos de graduação na área de computação. Technical report, Ministério da Educação, Brasília, DF. Disponível em: `https://www.gov.br/mec/pt-br/cne/arquivos/pdf/Res05.pdf`. Acesso em: 13 jul. 2025.

Castro, T., Castro Junior, A., Menezes, C., Boeres, M. C., and Rauber, M. C. (2002). Utilizando programação funcional em disciplinas introdutórias de computação. In *"Anais do WEI"*.

Hu, Z., Hughes, J., and Wang, M. (2015). How functional programming mattered. *National Science Review*, 2(3):349–370. Advance access publication 13 July 2015.

Hughes, J. (1990). Why functional programming matters. In Turner, D. A., editor, *Research Topics in Functional Programming*, pages 17–42. Addison-Wesley, Wokingham.

Khanfor, A. and Yang, Y. (2017). An overview of practical impacts of functional programming. In *2017 24th Asia-Pacific Software Engineering Conference Workshops (APSECW)*, pages 50–54.

Lucena, L. R. and Lucena, M. (2016). Potigol, a programming language for beginners. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, pages 368–368.

Pérez, Y. P. and López, L. M. (2007). Multiparadigma en la enseñanza de la programación. In *IX Workshop de Investigadores en Ciencias de la Computación*.

Potigol (2021). Potigol: Linguagem potigol – linguagem de programação funcional moderna para iniciantes. `https://github.com/potigol/potigol`. Acesso em: 9 julho 2025.

Potigol (2025). 800+ soluções de problemas do beecrowd usando a linguagem potigol. `https://github.com/potigol/beecrowd`. Acesso em: 9 julho 2025.