perplexity

Resumo do Texto sobre Padrões de Projeto

O texto apresenta uma introdução aos padrões de projeto em desenvolvimento de sistemas, destacando sua importância, principais categorias e exemplos práticos de aplicação.

Contexto e Origem

- Os padrões de projeto surgiram como uma adaptação dos conceitos de boas práticas da arquitetura, inicialmente propostos por Christopher Alexander nos anos 1970, para o desenvolvimento de software.
- Em 1994, quatro autores conhecidos como GoF (Gang of Four) - Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides - publicaram o primeiro catálogo de padrões de projeto voltados para software orientado a objetos, tornando-se referência mundial no tema.

Conceito

- Um padrão de projeto descreve soluções recorrentes para problemas comuns no desenvolvimento de software, permitindo reutilização de conhecimento e facilitando manutenção e evolução dos sistemas.
- O uso de padrões não significa copiar códigos prontos, mas sim adotar estruturas e práticas testadas, adaptando-as conforme a necessidade do projeto.

Categorias de Padrões de Projeto

Os padrões foram organizados pelo GoF em três grandes grupos:

- **Padrões Criacionais:** Focam na criação flexível de objetos, controlando como e quando instanciar classes. Exemplos: Abstract Factory, Factory Method, Singleton, Builder, Prototype.
- **Padrões Estruturais:** Facilitam a composição e organização de classes e objetos, ajudando a estruturar sistemas mais flexíveis. Exemplos: Proxy, Adapter, Facade, Decorator, Bridge, Composite, Flyweight.
- **Padrões Comportamentais:** Tratam da interação entre objetos e da distribuição de responsabilidades. Exemplos: Strategy, Observer, Template Method, Visitor, Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, State.

Exemplos Práticos

- **Abstract Factory:** Permite criar famílias de objetos relacionados sem especificar suas classes concretas, tornando o sistema mais flexível e fácil de manter. O texto exemplifica com um sistema de sugestão de jogos (esporte e RPG), mostrando como a implementação baseada em fábricas abstratas facilita a adição de novos tipos de jogos.
- **Factory Method:** Centraliza a criação de objetos em um único método, útil para situações mais simples que não requerem toda uma família de objetos. O exemplo dado é a criação de

diferentes polígonos (triângulo, retângulo, pentágono) conforme o número de lados informado.

- **Composite:** Utilizado para compor objetos em estruturas hierárquicas, como menus com itens e subitens, permitindo tratar objetos individuais e composições de forma uniforme.

Conclusão
O texto enfatiza que conhecer e aplicar os principais padrões de projeto é essencial para programadores que desejam criar sistemas mais robustos, flexíveis e fáceis de manter. Além disso, incentiva o estudo contínuo dos padrões para ampliar o repertório de soluções em desenvolvimento de software.

⁂

Assim como com os 12 princípios ágeis, existem 4 valores que compõem o manifesto ágil. Analise os valores abaixo:

I. Indivíduos e interações mais que processos e ferramentas
II. Documentação abrangente mais que software em funcionamento.
III. Colaboração com cliente mais que negociações de contratos.
IV. Responder a mudanças menos que seguir um plano.

Quais deles fazem parte do manifesto ágil:

```
a.
```

II e III.

```
b.
```

Todas as alternativas.

```
c.
```

I e III.

```
d.
```

II e IV.

```
e.
```

Nenhuma das alternativas.

Os quatro valores do Manifesto Ágil são:

1. Indivíduos e interações mais que processos e ferramentas
2. Software em funcionamento mais que documentação abrangente
3. Colaboração com o cliente mais que negociação de contratos
4. Responder a mudanças mais que seguir um plano[1] [2] [3].

Analisando as alternativas apresentadas:

- I. Indivíduos e interações mais que processos e ferramentas - **correto, faz parte do manifesto ágil**[1] [2] [3].
- II. Documentação abrangente mais que software em funcionamento - **incorreto, o valor correto é "software em funcionamento mais que documentação abrangente"**[1] [2] [3].
- III. Colaboração com cliente mais que negociações de contratos - **correto, faz parte do manifesto ágil**[1] [2] [3].
- IV. Responder a mudanças menos que seguir um plano - **incorreto, o valor correto é "responder a mudanças mais que seguir um plano"**[1] [2] [3].

Portanto, apenas as alternativas I e III correspondem aos valores do Manifesto Ágil.

Resposta correta:
c. I e III.

⁂

Sobre a prática TDD (desenvolvimento guiado por testes) é correto afirmar que:

```
a.
```

Testes devem ser realizados na elicitação do projeto.

```
b.
```

É possível obter uma clareza do que já foi produzido, no entanto essa prática precisa de um tempo para adaptação em sua aplicação.

```
c.
```

Ao implementar uma nova funcionalidade no sistema, os testes automatizados não garantem segurança para as que já existem.

```
d.
```

O refinamento é a etapa mais longa, pois é necessário atender a todas as necessidades.

```
e.
```

É levado menos tempo para criação dos testes, pois as funcionalidades são desenvolvidas primeiro.

A alternativa correta é:

b. É possível obter uma clareza do que já foi produzido, no entanto essa prática precisa de um tempo para adaptação em sua aplicação.

Veja a análise das opções:

- **a. Testes devem ser realizados na elicitação do projeto.**
  Incorreta. O TDD propõe que os testes sejam escritos antes do código de produção, mas durante o desenvolvimento, não na fase de elicitação de requisitos[4] [5] [6].
- **b. É possível obter uma clareza do que já foi produzido, no entanto essa prática precisa de um tempo para adaptação em sua aplicação.**
  Correta. O TDD fornece clareza sobre o que foi implementado, já que cada funcionalidade é coberta por testes automatizados, mas há uma curva de aprendizado e adaptação para quem começa a aplicar a prática[7] [8] [9].
- **c. Ao implementar uma nova funcionalidade no sistema, os testes automatizados não garantem segurança para as que já existem.**
  Incorreta. Um dos principais benefícios do TDD é justamente garantir que novas funcionalidades não quebrem as já existentes, pois os testes automatizados são executados continuamente para validar todo o sistema[5] [10] [6].
- **d. O refinamento é a etapa mais longa, pois é necessário atender a todas as necessidades.**
  Incorreta. A etapa de refatoração (refinement/refactor) é importante, mas não necessariamente a mais longa; ela visa melhorar o código sem alterar seu comportamento, sempre garantindo que os testes continuem passando[4] [10].
- **e. É levado menos tempo para criação dos testes, pois as funcionalidades são desenvolvidas primeiro.**
  Incorreta. No TDD, os testes são criados antes das funcionalidades, o que pode aumentar o tempo inicial de desenvolvimento, embora reduza retrabalho e erros a longo prazo[7] [9].

Portanto, a alternativa **b** é a correta.

⁂

Ao utilizar uma metodologia ágil, o projeto é dividido em pequenas partes, possibilitando a entrega ao cliente de versões funcionais, porém simplificadas do software, atendendo as necessidades mínimas para funcionamento. Esse conceito é conhecido como pela sigla MVP, que é o acrônimo de:

```
a.
```

Melhor Jogador da Partida.

```
b.
```

Maior Produto Viável.

```
c.
```

Mínimo Produto Viável.

```
d.
```

Menor Produto Viável.

```
e.
```

Máximo Produto Viável.

A resposta correta é:

c. Mínimo Produto Viável.

Explicação

MVP significa **Mínimo Produto Viável** (em inglês, Minimum Viable Product).
Esse conceito é muito utilizado em metodologias ágeis e refere-se à versão mais simples de um produto que pode ser lançada para o cliente, contendo apenas as funcionalidades essenciais para que ele funcione e gere valor ao usuário. O objetivo é validar ideias rapidamente, obter feedback e evoluir o produto de acordo com as necessidades reais do cliente.

As demais alternativas não correspondem ao significado de MVP nesse contexto.

⁂

O modelo cascata é uma metodologia aplicada em projetos onde o planejamento inicial era detalhado e seguido em sequência, com rica documentação. Porém percebeu-se que no desenvolvimento de software não era possível aplicar os mesmos métodos. Assim criou-se um novo método de controle de processo, conhecido como manifesto ágil, esse fato se deu pois:

```
a.
```

As aplicações sofrem várias mudanças ao longo do desenvolvimento.

```
b.
```

O modelo cascata já permitia entregas em ciclos curtos e iterativos de desenvolvimento.

```
c.
```

As aplicações precisavam ficar prontas de forma rápida, pois o cliente sabia exatamente o que desejava.

```
d.
```

Metodologias ágeis são conhecidas por processos engessados e com ciclo longo de desenvolvimento.

```
e.
```

As aplicações não sofrem mudanças ao longo do projeto.

A alternativa correta é:

a. As aplicações sofrem várias mudanças ao longo do desenvolvimento.

Explicação

O modelo cascata pressupõe um planejamento inicial detalhado e um desenvolvimento sequencial, com pouca flexibilidade para mudanças. No entanto, no desenvolvimento de software, percebeu-se que os requisitos e necessidades dos clientes mudam frequentemente ao longo do projeto. Por isso, o Manifesto Ágil foi criado, propondo métodos mais adaptáveis e iterativos, que permitem responder rapidamente a mudanças.

As demais alternativas estão incorretas porque:

- **b.** O modelo cascata **não** permite entregas em ciclos curtos e iterativos.
- **c.** Frequentemente o cliente **não sabe exatamente** o que deseja desde o início.
- **d.** Metodologias ágeis são conhecidas justamente pela flexibilidade e ciclos curtos.
- **e.** As aplicações **sofrem sim** mudanças ao longo do projeto.

Portanto, a alternativa **a** é a correta.

⁂

1. https://pt.wikipedia.org/wiki/Manifesto_Ágil
2. https://pt.smartsheet.com/comprehensive-guide-values-principles-agile-manifesto
3. https://tiflux.com/blog/manifesto-agil/
4. https://www.pontotel.com.br/desenvolvimento-orientado-a-testes/
5. https://www.impacta.com.br/blog/desenvolvimento-orientado-a-testes-tdd-vale-a-pena-investir-nessa-metodologia/
6. https://www.dio.me/articles/test-driven-development-tdd-desenvolvimento-guiado-por-testes
7. https://blog.elo7.dev/tdd/
8. https://chroniclesofapragmaticprogrammer.substack.com/p/o-que-estamos-perdendo-ao-ignorar-o-tdd
9. https://www.devmedia.com.br/test-driven-development-tdd-simples-e-pratico/18533
10. https://www.e-core.com/lt-pt/artigos/test-driven-development-na-pratica/