DES-SIS-II - A1 Texto de apoio

Site: EAD Mackenzie
Tema: DESENVOLVIMENTO DE SISTEMAS II {TURMA 03B} 2023/1
Livro: DES-SIS-II - A1 Texto de apoio

Impresso por: FELIPE BALDIM GUERRA .
Data: quarta, 1 mar 2023, 05:53

Descrição

Índice

1. PADRÕES: O QUE SÃO E PARA QUE SERVEM?

Padrões são importantes em muitas áreas do conhecimento. A existência de padrões ajuda em muitos aspectos do dia a dia, no sentido de padronizar coisas e procedimentos e estabelecer um vocabulário comum entre as pessoas.

Pense, por exemplo, em um carro. Todo carro é composto de muitas partes, que nem sempre são fabricadas no mesmo lugar, nem pelo mesmo fornecedor. Se as medidas e especificações dessas partes não seguissem padronizações, seria virtualmente impossível a montagem de qualquer veículo.

Figura 1 – Os diversos componentes de um carro

Fonte: Licença CC-BY 2.0.

Assim, muitas das especificações que vemos hoje, em muitos detalhes do dia a dia, são resultados da necessidade de padrões: tamanho de parafusos, entradas USB, formatos de tomadas, padrões de transmissão de TV digital, entre muitas outras. Estamos rodeados de padrões!

Figura 2 – Exemplos de padrões: padrão brasileiro de tomadas de 10A (à esquerda) e diferentes padrões de parafusos (à direita)

Fonte: Wikipedia; Wikipedia.

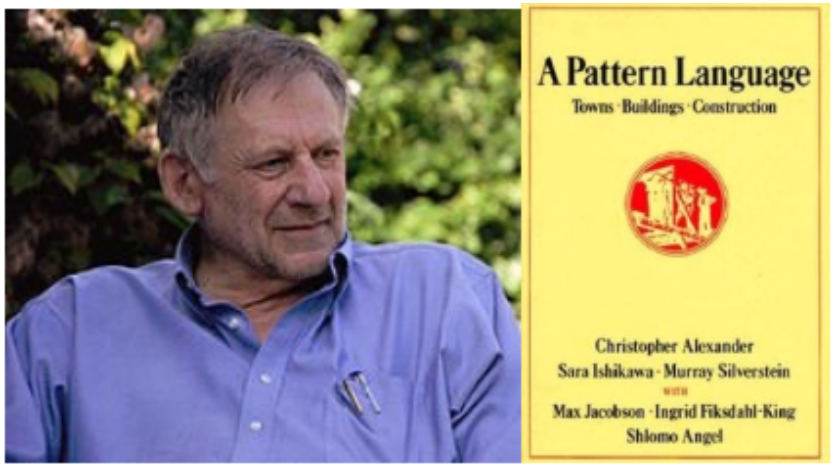
E para que precisamos de padrões? Resumidamente, precisamos de padrões porque:

- Padrões trazem soluções já testadas e com resultados conhecidos;
- Padrões fazem com que o desenvolvimento de produtos e processos seja mais seguro, mais eficiente e até mesmo mais barato;
- Padrões evitam discussões sobre “reinventar a roda”, fazendo com que o desenvolvimento de qualquer produto seja mais rápido;
- Padrões facilitam a integração entre elementos de fornecedores diferentes (o que conhecemos por “interoperabilidade”).

Mas, o que são padrões de software?

Por mais incrível que possa parecer, a definição mais ampla de padrões no contexto do desenvolvimento de software não veio de nenhum grande autor na área de TI. Ela veio de Christopher Alexander (abaixo, à esquerda), um professor de Arquitetura da Universidade da Califórnia, em Berkeley que, em 1977, em seu hoje famoso livro A Pattern Language (abaixo, à direita), junto com companheiros, estabeleceu definições para o que seria o conceito de padrões na área da Arquitetura.

Figura 3 – Christopher Alexander e seu livro



Fonte: Wikipedia; Wikipedia.

Entre outras coisas, Alexander definiu que:

- “Cada padrão é uma regra de três partes, que expressa uma relação entre um certo contexto, um problema e uma solução .”
- “Cada padrão descreve um problema em nosso ambiente e o núcleo de sua solução , de tal forma que você possa usar esta solução mais de um milhão de vezes , sem nunca fazê-lo da mesma maneira.”

Entenderemos a definição de padrões, partindo de um exemplo da área da qual ela veio: a Arquitetura.

Você deve conhecer o Museu de Arte de São Paulo (MASP), ao menos por fotografia.

Figura 4 – O prédio do MASP

Fonte: Wikipedia.

Projetado pela arquiteta ítalo-brasileira Lina Bo Bardi, sua construção foi finalizada em 1968. O prédio ficou conhecido por seu vão livre de 74 metros que se estende sob quatro pilares enormes e chamativos, pintados de vermelho. O pavilhão, visível da Avenida Paulista, é, na verdade, um grande bloco suspenso a oito metros de altura do piso, segurado pelos pilares.

O que pouca gente sabe é que essa obra-prima da arquitetura tem uma razão de ser: antigamente, ali existia um belvedere , espaço de encontro da elite paulistana da época, e que tinha uma bela vista para o centro da cidade. O terreno foi doado para a construção do MASP, com a condição de que a vista fosse mantida.

Figura 5 – Belvedere Trianon em 1930 (ele foi demolido em 1951)

Fonte: Wikipedia.

A arquiteta Lina, para preservar a vista para o centro da cidade, tinha apenas duas opções: uma edificação subterrânea ou uma suspensa. E ela escolheu as duas. Hoje, o MASP é composto por um bloco subterrâneo e um elevado.

- Você consegue enxergar o contexto , o problema e a solução encontrada por Lina?
- Você acha que essa solução de Lina Bo Bardi pode ser reaplicada em um outro contexto que apresente um problema similar?

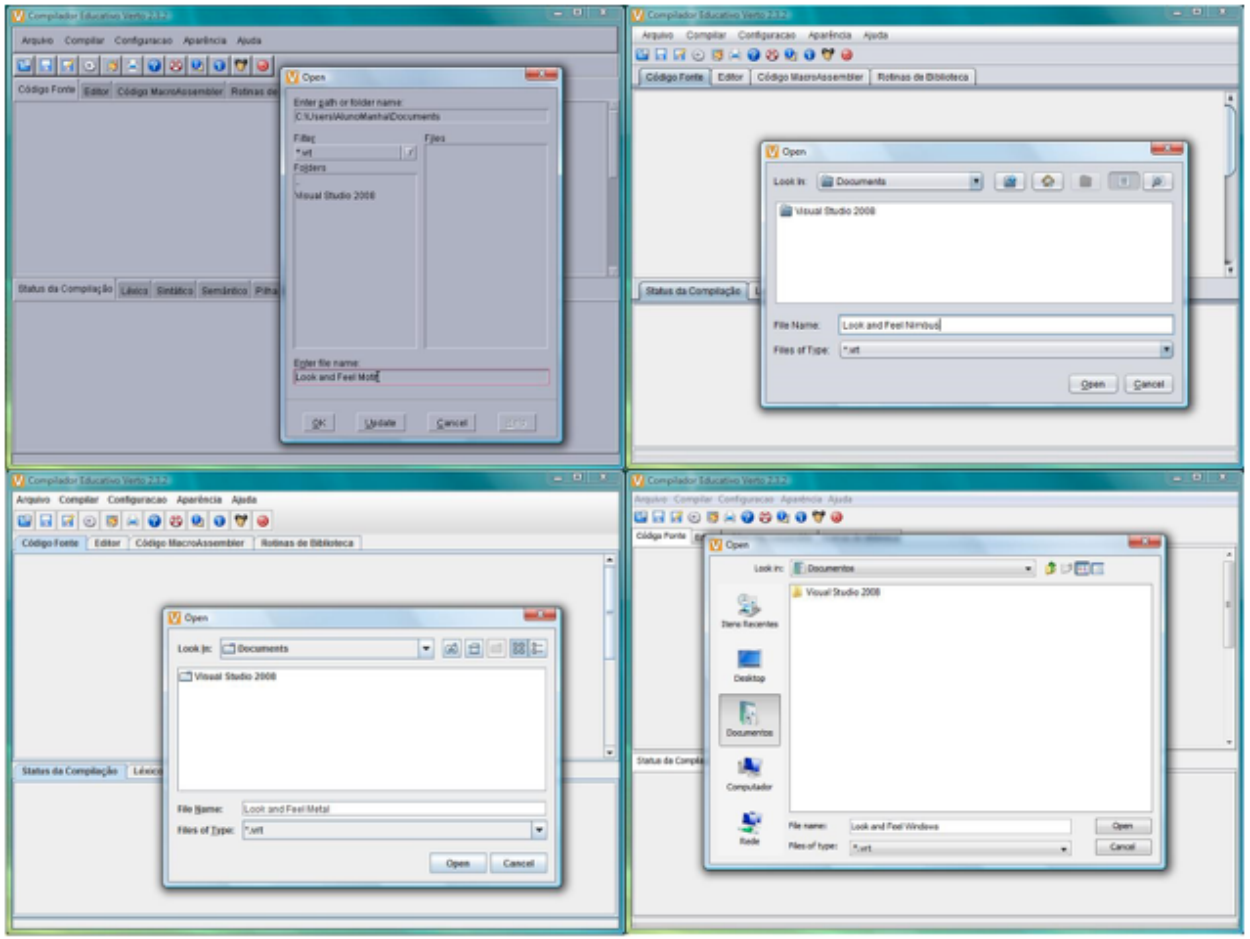
Se você respondeu SIM a ambas as perguntas, então está frente a um bom candidato para se tornar um padrão na Arquitetura!

Níveis de padronização no desenvolvimento de software

E em nossa área de desenvolvimento de softwares? Por vezes, é difícil imaginar padrões – especialmente se nos limitarmos às interfaces de diferentes aplicativos, quando executados em distintos Sistemas Operacionais, por exemplo. Mas saiba que, ainda que visualmente possa haver diferenças (que chamamos de look-and-feel), muito do comportamento dessas interfaces é padronizado. Porém, nossa ideia aqui é ir muito além das interfaces!

Na figura seguinte, vemos um mesmo software desenvolvido em Java com diferentes look-and-feel para quatro Sistemas Operacionais de desktop diferentes. Parece despadronizado, mas o comportamento dos elementos das interfaces tem muitas similaridades.

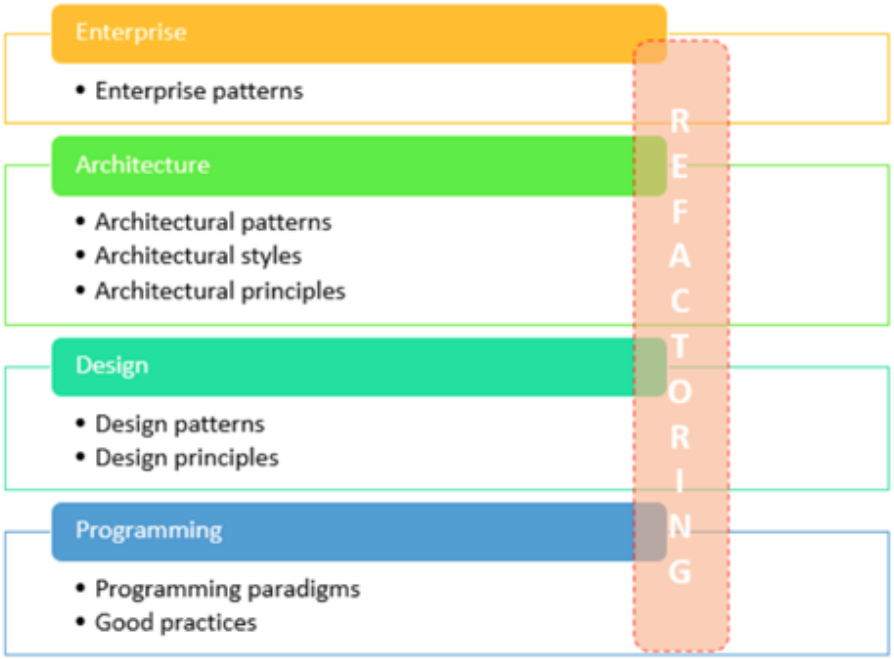
Figura 6 – Software desenvolvido em Java com diferentes look-and-feel



Fonte: Wikipedia.

Há muitos níveis de abstração quando falamos em padrões no processo de desenvolvimento de software. A figura a seguir mostra uma visão esquemática a respeito dos diferentes níveis de padronização possíveis (os nomes estão em inglês, pois eles são geralmente mencionados assim no mercado e na Academia, geralmente sem tradução):

Figura 7 – Níveis de abstração em padrões de desenvolvimento de software
Fonte: Elaborado pelo autor.



Falaremos um pouco sobre cada um desses níveis.

> Não se preocupe muito agora com os detalhamentos de cada nível, pois eles serão aprofundados no decorrer da disciplina.

Nível Enterprise:

Quando falamos de padrões empresariais (ou padrões arquiteturais empresariais), estamos nos referindo a padrões que buscam ajudar no planejamento do panorama tecnológico e organizacional de uma empresa e em como integrar as soluções de TI nas organizações. Diferente do software tido como "convencional" (um aplicativo voltado a um usuário final, que roda em desktop/notebook ou mobile), as aplicações empresariais geralmente são caracterizadas por demandarem grandes volumes de processamento ou de manipulação e gerenciamento de dados. Elas comumente dependem de arquiteturas com alto grau de escalabilidade e distribuição (como clusters e nuvem, por exemplo) e o planejamento da integração dessas soluções com o core business das organizações é crucial. Assim, os padrões tornam-se importantes nesse contexto de mais alto nível.

Nível Architecture:

Este é um nível de abstração crucial para virtualmente qualquer software, seja ele um grande sistema de informação empresarial ou uma solução de um app mobile. Há diversos catálogos com **Padrões Arquiteturais**, mas também com **Estilos** e **Princípios** arquiteturais – aqui, estamos falando de arquitetura de software e não mais da área de trabalho da Lina Bo Bardi...

Não se preocupe muito, neste momento, em estabelecer diferenças cruciais entre padrões, estilos e princípios quando falamos em Arquitetura de Software – muitos autores misturam essas definições – mas, a princípio, você pode considerar os **padrões** arquiteturais como um conjunto mais consistente de contexto/problema/solução, enquanto os **estilos** são um conceito um pouco mais abstrato, mais ligado à solução, definindo como cada um dos elementos da arquitetura vão se comportar e interagir. Já os **princípios**, dizem respeito às boas práticas e regras gerais que podem ser inerentes a cada arquitetura ou que podem ser de cunho geral. Veja a tabela a seguir com alguns exemplos de padrões, estilos e princípios arquiteturais. Novamente, os termos são apresentados em inglês, porque geralmente não são traduzidos.

TABELA

Padrões Arquiteturais	**Estilos Arquiteturais**	**Princípios Arquiteturais**

Padrões Arquiteturais	**Estilos Arquiteturais**	**Princípios Arquiteturais**
Component-based	Database-centric	Solid principles
Domain-driven design	Monolithic application	Single Responsibility
Client/server	Layered	Open-Closed
Distributed P2P	Pipes and filters	Liskov Substitution
computing	Event-driven	Interface Segregation
Message bus	Publish-subscribe	Dependency Inversion
Model-view-controller	Plug-ins	Principles of economics
Microkernel		Opportunity Cost
Blackboard		Cost of Delay
Service-oriented		Principles of least
Microservices		Least Astonishment
Space-based		Least Effort

> Não se preocupe em memorizar os nomes nem em se aprofundar sobre cada um deles – no decorrer do componente curricular, estudaremos vários deles.

Nível Design:

Um padrão de arquitetura expressa uma proposta de organização ou uma estrutura esquemática em alto nível para sistemas computacionais complexos. Já um padrão de projeto (design) é baseado em uma estrutura recorrente de comunicação entre componentes de um software, que resolvem um problema geral dentro de um determinado contexto, em um nível mais detalhado, próximo à implementação. Neste nível, nos aprofundaremos, no decorrer do componente curricular, em Princípios de Design do catálogo GRASP (General Responsibility Assignment Software Patterns) e Padrões de Projeto (Design Patterns) do catálogo GoF (Gang of Four). Como de hábito, são mantidos os nomes em inglês para ambos.

Um padrão de arquitetura expressa uma proposta de organização ou uma estrutura esquemática em alto nível para sistemas computacionais complexos. Já um padrão de projeto (design) é baseado em uma estrutura recorrente de comunicação entre componentes de um software, que resolvem um problema geral dentro de um determinado contexto, em um nível mais detalhado, próximo à implementação. Neste nível, nos aprofundaremos, no decorrer do componente curricular, em Princípios de Design do catálogo GRASP (General Responsibility Assignment Software Patterns) e Padrões de Projeto (Design Patterns) do catálogo GoF (Gang of Four). Como de hábito, são mantidos os nomes em inglês para ambos.

TABELA

Princípios de Design GRASP	**Padrões de Projeto GoF**

Princípios de Design GRASP	**Padrões de Projeto GoF**
Creator	**Criacionais:**
Controller	Abstract Factory
Indirection	Builder
Information expert	Factory Method
Low coupling	Prototype
High cohesion	Singleton
Polymorphism	**Estruturais:**
Protected variations	Adapter
Pure fabrication	Bridge
	Composite
	Decorator
	Façade
	Flyweight
	Proxy
	Comportamentais:
	Chain of Responsibility
	Command
	Interpreter
	Iterator
	Mediator

Princípios de Design GRASP	Padrões de Projeto GoF
	Memento Observer State Strategy Template Method Visitor

Nível Programming:

Neste nível, a padronização se dá tanto nos diferentes paradigmas (modelos) de programação (Orientação a Objetos, Procedimental, Funcional etc.) quanto nas linguagens de programação em si. O desenvolvimento de código deve também se guiar por boas práticas de programação.

Nesse contexto, a não aplicação de boas práticas pode levar o código desenvolvido a apresentar o que se convencionou chamar de "mau cheiro" (bad smell): não são exatamente trechos incorretos (do ponto de vista de funcionalidade), mas que foram construídos de maneira a não facilitar o processo de manutenção, documentação, compreensão ou modificação do código – ou ainda torná-lo menos eficiente ou expansível. Nesse ponto, entra a **Refatoração**,

que é uma técnica disciplinada para reestruturar um código existente (em partes ou no todo), alterando sua estrutura interna sem mudar seu comportamento externo. Consiste, então, de uma série de pequenas transformações que preservam o comportamento observável do código, melhorando os aspectos mencionados acima. Existe um extenso catálogo de refatorações em < https://refactoring.com/catalog/>

Importante mencionar que o princípio básico da refatoração ("alterando sua estrutura interna sem mudar seu comportamento externo"), embora tenha originalmente surgido no contexto do código (nível programming), pode ser aplicado também no nível design (como pode ser visto no catálogo <https://industriallogic.com/xp/refactoring/catalog.html >) ou mesmo em níveis de maior abstração, como no nível architecture ou até mesmo no nível enterprise.