Faculdade XPe

Capítulo 07

ESTILOS ARQUITETURAIS

Gladston Junio Aparecido

2024

Estilos Arquiteturais

A concepção da arquitetura de um software é uma equação complexa que envolve inúmeras variáveis, oriundas, principalmente, dos requisitos esperados para o sistema. Esse processo demanda a análise e ponderação dos trade offs de cada possível abordagem candidata e o arquiteto deve ser capaz de avaliá–las em um alto nível de abstração, uma vez que o sistema ou componente está em fase preliminar de concepção.

Se o arquiteto tiver em mãos diretrizes que permitem organizar as abordagens candidatas de forma a reconhecer suas características e permitir determinar se o sistema alvo é semelhante a outros no qual cada abordagem já foi aplicada ele será capaz de avaliar os benefícios e os pontos negativos e, assim, fundamentar suas decisões. Porém, as soluções precisam ser abstratas o suficiente de forma a viabilizar a comparação independente de tecnologias e ramo de atuação de negócio.

Tudo isso torna fundamental o conhecimento dos estilos arquiteturais. Estilos arquiteturais descrevem soluções para resolução de problemas na arquitetura de software, porém, atuam em alto nível, permitindo caracterizar uma família de sistemas do ponto de vista da sua organização estrutural. Isso significa que eles produzem uma terminologia que caracteriza componentes de software com base no aspecto identificável da sua organização, sempre a partir de uma visão do nível mais alto de abstração e independente de tecnologia. O estilo arquitetural de um software permite descrever como as camadas, os componentes e os módulos irão se comunicar, identificando e isolando suas responsabilidades.

Não existe um estilo arquitetural universal capaz de resolver todos os problemas, uma vez que cada tipo de sistema possui uma característica marcante. Por isso, cada estilo arquitetural é capaz de resolver uma classe de problemas. Existem sistemas que demandam requisitos elevados de segurança, enquanto outros podem ter como característica fundamental o desempenho, a replicação global de dados ou um elevado índice de disponibilidade. Desta forma, os sistemas que seguirem um determinado estilo arquitetural se beneficiarão de determinados atributos em favor de outros.

É importante salientar que estilos arquiteturais não são sinônimos de padrões arquiteturais. Um padrão arquitetural apresenta uma solução amplamente testada, documentada e replicável para problemas recorrentes de arquitetura dentro de um estilo arquitetural. Esses termos são frequentemente confundidos, porém, atuam em níveis distintos da solução. Estilos arquiteturais fornecem uma visão periférica em torno do problema enquanto padrões arquiteturais descrevem o processo de como a solução deve ser construída, focado em uma solução concreta.

As seções a seguir detalham os principais tipos de estilos arquiteturais.

Estilos Arquiteturais – Structure

Os estilos arquiteturais do tipo Structure têm por objetivo suportar o planejamento da arquitetura do sistema fornecendo diretrizes para a decomposição e organização dos componentes e comportamentos do sistema. Também são conhecidos pelo jargão “From mud to structure”. Os principais são:

- Layered
- Pipes and filters
- Component-based

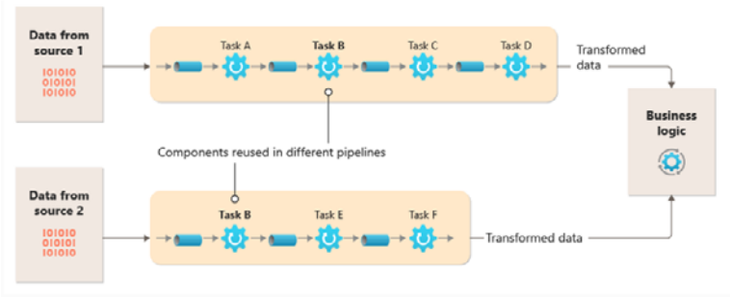
Esses estilos têm alta relevância na teoria de arquitetura de software uma vez que as discussões sobre como distribuir as responsabilidades fazem parte de todos sistemas minimamente estruturados. Em contrapartida, nem todos sistemas demandam requisitos de adaptação, por exemplo.

O estilo arquitetural Layered, também conhecido como arquitetura baseada em camadas, sugere a organização dos sistemas em cada camada, em que cada camada possui uma responsabilidade específica e todos os artefatos de software residentes em uma determinada camada devem desempenhar um papel associado a essa responsabilidade. Seu uso contribui para desenvolver aplicações mais fáceis de entender, testar e manter, além de auxiliar o desenvolver paralelo e independente de cada camada do sistema. O estilo arquitetural por si não define de forma explícita as camadas que devem compor o sistema, ele é frequentemente refinado e transformado em uma solução concreta pelos padrões arquiteturais como MVC, MVP, dentre outros que serão estudados nos capítulos posteriores do curso. Entretanto, é comum encontramos pelo menos quatro camadas nas principais soluções:

1. Presentation Layer,
2. Business Layer,
3. Persistence Layer
4. Database Layer.

Já o estilo arquitetural Pipe and Filters, ilustrado em Figura 8, visa decompor um processo complexo de forma que cada tarefa é encapsulada em elementos distintos e reutilizados e, a execução do processo como um todo ocorre através do encadeamento de tais componentes. Além de permitir o reuso das atividades, esse modelo arquitetural contribui para a melhoria do desempenho dos processos e simplifica o processo evolutivo, permitindo que novas etapas sejam facilmente inseridas ou removidas do fluxo de processamento.

Figura 8 – Padrão Registry



Fonte: https://learn.microsoft.com/pt-br/azure/architecture/patterns/pipes-and-filters

As premissas do estilo Pipe and Filters são utilizadas na organização de componentes autônomos de sistemas distribuídos, no processamento de requisições HTTP de servidores web e também na organização de camadas em alguns padrões arquiteturais que estudaremos nos capítulos posteriores, como Hexagonal Architecture.

Estilos Arquiteturais – Message Styles

O processamento assíncrono entre sistemas distribuídos pode ser utilizado para garantir a conformidade com diferentes requisitos de qualidade de um sistema que, via integração síncrona, podem não ser tecnicamente viáveis. Os estilos arquiteturais deste tipo classificam as diferentes abordagens frequentemente adotadas no uso de processamento de mensagens, sendo os principais:

- Implicit invocation
- Asynchronous messaging
- Publish-subscribe

Esses três estilos arquiteturais formam a base das principais arquiteturas modernas, em que a presença de middlewares orientados a mensagens nas integrações entre aplicações e componentes da própria aplicação e o uso de eventos para processamento assíncrono estão cada vez mais presentes.

Estilos Arquiteturais – Adaptive Systems

Existem cenários que imputam aos sistemas a necessidade de serem flexíveis e customizáveis, permitindo que mudanças nos requisitos funcionais e não funcionais sejam aplicadas diretamente pelos consumidores do sistema. Os estilos arquiteturais deste tipo categorizam as principais soluções adotadas para satisfazer essa necessidade. Os principais estilos arquiteturais catalogados nesta categoria são:

- Microkernel
- Reflection
- Domain-specific language

Estilos Arquiteturais – Distributed Systems

Um arquiteto deve dominar as boas práticas de projeto e desenvolvimento de sistemas distribuídos e os estilos arquiteturais desta categoria permitem identificar as características chaves das principais abordagens para composição de sistemas distribuídos e integração entre aplicações. Os principais estilos arquiteturais desta categoria são:

- Service-oriented
- Peer to peer style
- Object request broker
- Cloud native.

Estilos Arquiteturais – Deployments

A arquitetura é uma disciplina holística e envolve as diversas fases do ciclo de vida de um software e suas diferentes necessidades. Saber classificar e analisar o trade offs dos principais modelos de implantação dos sistemas é fundamental para o sucesso de um software, uma vez que a escolha do modelo de implantação incorreto pode anular todos os pontos positivos do projeto de um software. Os estilos arquiteturais mais conhecidos deste tipo são:

- Client-Server
- N-tier

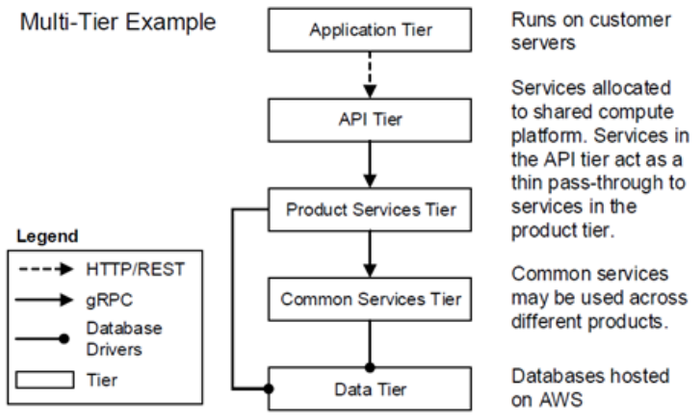
Conforme ilustra a Figura 9, no estilo arquitetural Client-Server visa distribuir o runtime do software é distribuído entre dois tipos de nós e ambos se integram por meio de um canal de comunicação. Os componentes presentes nos clientes podem, ainda, serem estruturados com base em uma separação de camadas lógicas (layers), assim como no servidor. Além disso, um cliente pode se comunicar com um ou vários servidores diferentes utilizando vários protocolos.

Figura 9 – Arquitetura Cliente Servidor

Um cliente pode requisitar dados e algoritmos para um servidor. Além disso, os clientes podem ser classificados como Thin Clients e Fat Clients. O critério para essa definição se baseia na capacidade de processamento e quantidade de responsabilidades atribuídas para os componentes de software que rodam no cliente.

A arquitetura N-tier se baseia nos princípios do estilo arquitetural N-tier, porém, tem foco na distribuição física do software (tiers). Seu principal objetivo é distribuir o software em camadas físicas de acordo com a complexidade e os requisitos arquiteturais do software. Embora a implementação mais tradicional seja a separação em três camadas, nas aplicações modernas é comum a utilização de várias camadas como pode ser observado no exemplo ilustrado na Figura 10.

Figura 10 – Arquitetura N-Tier



Fonte: KEELING (2017).

Um sistema pode demandar diversos estilos arquiteturais para atender às suas necessidades, além disso, eles podem ser combinados, produzindo soluções híbridas. Este capítulo descreve brevemente os estilos arquiteturais e devem ser utilizados como um guia para o aprofundamento dos estudos.

Referências

BROWN, William J. et al. AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis. 1. ed. Wiley, 2008. 336 p.

BUSCHMANN, Frank; SOMERLAD, Peter. Pattern-Oriented Software Architecture, a System of Patterns: Volume 1. 1. ed. Wiley, 1996. 476 p.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. Sistemas Distribuídos: conceitos e projeto. 4. ed. Porto Alegre: Bookman, 2007. ISBN: 9788560031498.

FOWLER, Martin. Patterns of Enterprise Application Architecture. 1. ed. Addison-Wesley Professional, 2002. 560 p.

FOWLER, Martin. Refactoring: Improving the Design of Existing Code. 1. ed. Addison-Wesley Professional, 2018. 448 p.

FREEMAN, Eric. Head First - Design Patterns. 1. ed. O'Reilly, 2004. 638 p.

GAMMA, Erich et al. Design Patterns: Elements of Reusable Object-Oriented Software. 1. ed. Addison-Wesley Professional, 1994. 416 p.

HALL, Gary M. Adaptive Code via C#: Agile coding with design patterns and SOLID principles. 1. ed. Microsoft Press, 2014. 401 p.

HOHPE, Gregor. The Software Architect Elevator: Redefining the Architect's Role in the Digital Enterprise. 1. ed. O'Reilly, 2020. 350 p.

HOHPE, Gregor; WOOLF, Bobby. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. 1. ed. Addison-Wesley Professional, 2003. 736 p.

INGENO, Joseph. Software Architect's Handbook: Become a successful software architect by implementing effective architecture concepts. 1. ed. Packt Publishing, 2018. 594 p.

JOSHI, Bipin. Beginning SOLID Principles and Design Patterns for ASP.NET Developers. 1. ed. Apress, 2016. 420 p.

KEELING, Michael. Design It!: From Programmer to Software Architect. 1. ed. O'Reilly, 2017. 354 p.

Referências

KRAFGIZ, Dirk; BANKE, Kalr; SLAMA, Dirk. Enterprise SOA: Service-Oriented Architecture Best Practices. 1. ed. Editora Prentice Hall, 2004. 408 p.

MALVEAU, Raphael C. et al. AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis. 1. ed. Wiley, 1998. 336 p.

MALVEAU, Raphael C.; MOWBRAY Thomas J. Software Architect Bootcamp. 1. ed. Prentice Hall, 2000. 352 p.

MARTIN, Robert C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. 1. ed. Pearson, 2017. 432 p.

MIKOWSKI, Michael; POWELL, Josh. Single Page Web Applications: JavaScript end-to-end. 1. ed. Greenwich:Manning Publications, 2013. 432 p.

MILLET, Scott; TUNE, Nick. Patterns, Principles and Practices of Domain-Driven Design. 1. ed. Wrox, 2014. 800 p.

MONSON-HAEFEL, Richard. 97 Things Every Software Architect Should Know: Collective Wisdom from the Expert. 1. ed. O'Reilly, 2009. 220 p.

PAI, Praseed; XAVIER, Shine. .NET Design Patterns. 1. ed. Packt Publishing, 2017. 314 p.

SHALLOWAY, Alan; TROTT, James R. Design Patterns Explained: A New Perspective on Object Oriented Design. 2. ed. Addison-Wesley Professional, 2014. 480 p.

SHYAM, Seshadri; BRAD, Green. AngularJS: Up and Running: Enhanced Productivity with Structured Web Apps Paperback. 1. ed. Sebastopol:O'Reilly Media, 2014. 275 p.

TANENBAUM, Andrew S.; STEEN, Maarten Van. Sistemas Distribuídos: princípios e paradigmas. 2. ed. Pearson/Prentice-Hall, 2007. ISBN: 9788576051428.

Faculdade
XPe

xpeducacao.com.br