Arquitetura de Software

O termo arquitetura de software surgiu no final da década de 60 e início da década de 70 com **Edsger Dijkstra e David Parnas**, embora se tornando mais difundida nos anos 90.

Podemos definir a arquitetura como a organização fundamental de um sistema incorporada em seus componentes, seus relacionamentos com o ambiente, e os princípios que conduzem seu design e evolução. **ISO/IEEE 1471-[2000].**

Outra definição de arquitetura de software é uma estrutura (ou estruturas) do sistema, a qual é composta de elementos de software, das propriedades externamente visíveis desses elementos, e dos relacionamentos entre eles; é a abstração do sistema. **Bass [98] 2003.**

Também podemos conceituar como um conjunto de componentes e seus relacionamentos, que deve satisfazer os requisitos funcionais e não funcionais do sistema. **Jazayere et al [2000].**

A arquitetura define o que é o sistema em termos de componentes computacionais e, os relacionamentos entre estes componentes, os padrões que guiam a sua composição e restrições. **Mary Shaw e D. Garlan [96].**

Em An Introduction to Software Architecture, David Garlan e Mary Shaw sugerem que a arquitetura de software é um nível de design voltado para questões que vão: "além dos algoritmos e das estruturas de dados da computação. A projeção e a especificação da estrutura geral do sistema emergem como um novo tipo de problema. As questões estruturais incluem organização total e estrutura de controle global; protocolos de comunicação, sincronização e acesso a dados; atribuição de funcionalidade a elementos de design; distribuição física; composição de elementos de design; escalonamento e desempenho; e seleção entre as alternativas de design." [[GAR93](http://www.wthreex.com/rup/process/referenc.htm#GAR93)]

As visões capturam as principais decisões de design estruturais mostrando como a arquitetura de software é decomposta em [componentes](http://www.wthreex.com/rup/process/artifact/ar_comp.htm) e como os componentes são ligados pelos **conectores** para gerar **formulários** úteis [[PW92](http://www.wthreex.com/rup/process/referenc.htm#PW92)]. Essas alternativas de design devem estar associadas aos [requisitos](http://www.wthreex.com/rup/process/workflow/requirem/co_req.htm), tanto funcionais quanto suplementares, e a outras restrições. Essas alternativas, por sua vez, impõem **restrições** adicionais aos requisitos e às futuras decisões de design em um nível inferior.

Sendo assim, arquitetura é composição dos elementos / componentes que compõem uma estrutura e como eles se relacionam. Identificar estes componentes e suas interações, das dependências de construção e tecnologias que apoiam a implementação.

O termo de arquitetura de software é utilizado para definir processo e produto.

O produto é o software propriamente dito, seria a representação da sua estrutura, utiliza-se da área de engenharia de software para tratar de produzir as estruturas de software, buscando reduzir a complexidade.

**Importância da Arquitetura**

Sua abstração serve como apoio para criar um entendimento geral de comunicação entre os papéis de para a construção do software.

O arquiteto, define a arquitetura de software: modelando o processo de negócio; planejando a estratégia das releases, requisitos de cada versão, planejamento de integração contínua, etc

Sua representação serve como guia para o projeto de sua implementação, teste e implantação do software.

Portanto, a vida útil do software depende de uma boa construção de arquitetura que permita realizar modificações, e sua evolução. Desta forma, podemos incluir novas funcionalidades sem causar impactos nas funcionalidades já existentes. O software pode sofrer alterações de forma pontual, sem afetar outros módulos.

**Termos Relacionados**

* Model Driven Design
* Maior Interoperabilidade: Orientação a serviços / Utilização de Agentes / Sistemas Heterogêneos (Várias linguagens)
* Virtualização
* Processos
* Computação nas Nuvens
* Web 2.0

**Estilos Arquiteturais**

O estilo da arquitetura define um padrão de uniformidade na arquitetura. Podendo ser definido por um conjunto de padrões, componentes, conectores específicos que funcionarão como elementos estratégicos para a construção do software.

Estes padrões são soluções para problemas específicos que ocorrem de forma recorrente em um determinado contexto que foram identificados a partir da experiência coletiva de desenvolvedores de software. A proposta original de padrões veio do trabalho de Christopher Alexander na área de arquitetura (para construções). Sua definição para padrões: 

*Cada padrão é uma regra (esquema) de três partes que expressa uma relação entre um certo contexto, um problema, e uma solução.*

O **contexto** descreve uma situação no desenvolvimento na qual existe um problema.

O **problema**, que ocorre repetidamente no contexto, deve também ser descrito bem como as **forças** (requisitos, restrições e propriedades) associadas a ele.

A **solução** descreve uma configuração ou estrutura de componentes e suas interconexões, obedecendo às forças do problema.

As **forças**, denominação dada por [Alexander], descrevem os requisitos que caracterizam o problema e que a solução deve satisfazer, as restrições que devem ser aplicadas às soluções e propriedades desejáveis que a solução deve ter.

Os estilos de Arquitetura – expressam esquemas de organização estrutural de sistemas, fornecendo um conjunto de componentes do sistema, suas responsabilidades e a forma de interação entre eles. – Cada estilo de arquitetura lida com diferentes tipos de atributos de qualidade.

**Algumas considerações dos estilos Arquiteturais:**

• A arquitetura de um sistema pode aderir a um ou mais estilos arquiteturais. – Um estilo define os tipos de elementos que podem aparecer em uma arquitetura e as regras que regem a interconexão entre estes elementos.

• Esses estilos podem simplificar o problema de definição de arquiteturas de sistema.

• A maioria dos sistemas de grande porte adere a vários estilos

• Estilos arquiteturais = “modelos arquiteturais”

**Exemplos de Estilos Arquiteturais**

• Cliente-Servidor

• Camadas

• Filtros e dutos (pipes and filters)

• Repositórios

• Orientado a eventos (publisher/subscriber)

• Objetos distribuídos, etc

**Representação Arquitetural – Arquitetura de Software**

|  |  |
| --- | --- |
| **Modelo de Arquitetura de Software** | |
| 7 | Software |
| 6 | Estrutura de Dados |
| 5 | Conectores |
| 4 | Ferramentas (Linguagens) |
| 3 | Regra de Negócios (Requisitos Funcionais) |
| 2 | Estilos / Padrões Arquitetura |
| 1 | Processos |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Modelo de Arquitetura de Software**  **Requisitos não Funcionais** | | | 7 | Manutenibilidade | | 6 | Confiabilidade | | 5 | Segurança | | 4 | Desempenho | | 3 | Escalável | | 2 | Disponibilidade | | 1 | Portabilidade | | |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Padrões Arquiteturais**

[[BUS96]](http://www.cin.ufpe.br/~rls2/processo_tg/openup/guidances/supportingmaterials/references_6CCF393.html#BUS96) agrupa os padrões arquiteturais de acordo com as características dos sistemas nos quais eles são mais aplicáveis, com uma categoria que trata das questões estruturais mais gerais. A tabela a seguir mostra as categorias apresentadas em [[BUS96]](http://www.cin.ufpe.br/~rls2/processo_tg/openup/guidances/supportingmaterials/references_6CCF393.html#BUS96) e os padrões que elas contêm.

|  |  |
| --- | --- |
| **Categoria** | **Padrões** |
| Estrutura | Camadas / Pipes e Filtros / Quadro-Negro |
| Sistemas Distribuídos | Agente |
| Sistemas Interativos | Modelo-Visão-Controle;  Apresentação-Abstração- Controle |
| Sistemas Adaptáveis | Reflexão / Microkernel |

**Enfoque da Arquitetura**

Embora as visões acima possam representar todo o design de um sistema, a arquitetura se preocupa somente com alguns aspectos específicos:

* A **estrutura** do modelo — os padrões organizacionais, por exemplo, a [divisão em camadas](http://www.wthreex.com/rup/process/workflow/ana_desi/co_layer.htm).
* Os **elementos essenciais** — os [casos de uso](http://www.wthreex.com/rup/process/artifact/ar_uc.htm) críticos, as [classes](http://www.wthreex.com/rup/process/artifact/ar_class.htm) principais, os mecanismos comuns, etc., em oposição a todos os elementos presentes no modelo.
* Alguns **cenários-chave** mostrando os principais fluxos de controle em todo o sistema.
* Os **serviços**, para capturar a modularidade, os recursos opcionais, os aspectos de linha de produto.

Basicamente, as visões de arquitetura são abstrações ou simplificações do design inteiro, em que as características importantes são ressaltadas, deixando os detalhes de lado. Essas características serão importantes quando os seguintes aspectos estiverem em discussão:

* Evolução do sistema — passagem para o próximo ciclo de desenvolvimento.
* Reutilização da arquitetura, ou de partes dela, no contexto de uma linha de produto.
* Avaliação das qualidades suplementares, como desempenho, disponibilidade, portabilidade e segurança.
* Atribuição do trabalho de desenvolvimento a equipes ou subcontratantes.
* Decisões sobre a inclusão dos componentes desenvolvidos internamente e adquiridos prontos para serem usados.
* Inserção em um sistema mais amplo.

Uma arquitetura de software, ou somente uma visão de arquitetura, pode ter um atributo chamado **estilo de arquitetura**, que reduz o conjunto de formulários que podem ser escolhidos e impõe um determinado grau de uniformidade à arquitetura. O estilo pode ser definido por um conjunto de padrões, ou pela escolha de componentes ou conectores específicos que funcionarão como os tijolos básicos da construção. Em um determinado sistema, alguns estilos podem ser capturados como parte da descrição da arquitetura em um **guia de estilo de arquitetura** — parte de um documento de [guia de design](http://www.wthreex.com/rup/process/artifact/ar_desgl.htm) no RUP. O estilo desempenha um papel vital na compreensão e integridade da arquitetura.

A representação gráfica de uma visão de arquitetura é denominada planta da arquitetura. Nas várias visões descritas acima, as plantas são compostas pelos seguintes diagramas da Unified Modeling Language [[UML99](http://www.wthreex.com/rup/process/referenc.htm#UML99)]:

* [**Visão lógica**](http://www.wthreex.com/rup/process/workflow/ana_desi/co_lview.htm). Diagramas de classe, máquinas de estado e diagramas de objetos
* [**Visão de processos**](http://www.wthreex.com/rup/process/workflow/ana_desi/co_pview.htm). Diagramas de classes e diagramas de objetos (tarefa abrangente — processos e threads)
* [**Visão de implementação**](http://www.wthreex.com/rup/process/workflow/implemen/co_impvw.htm). Diagrama de componentes
* [**Visão de implantação**](http://www.wthreex.com/rup/process/workflow/ana_desi/co_dview.htm). Diagramas de implantação
* [**Visão de casos de uso**](http://www.wthreex.com/rup/process/workflow/requirem/co_ucv.htm). Os diagramas de casos de uso representam casos de uso, atores e classes de design normais. Os diagramas de seqüências representam objetos de design e suas colaborações.

**Exemplos de arquitetura de software**

Há muitas formas comuns de projetar módulos de software de computador e suas comunicações, entre elas:

* [Cliente-Servidor](https://pt.wikipedia.org/wiki/Cliente-Servidor)
* [Computação distribuída](https://pt.wikipedia.org/wiki/Computa%C3%A7%C3%A3o_distribu%C3%ADda)
* [P2P](https://pt.wikipedia.org/wiki/P2P)
* [Quadro Negro](https://pt.wikipedia.org/wiki/Quadro_Negro_(computa%C3%A7%C3%A3o))
* [Criação implícita](https://pt.wikipedia.org/w/index.php?title=Cria%C3%A7%C3%A3o_impl%C3%ADcita&action=edit&redlink=1)
* [Pipes e filtros](https://pt.wikipedia.org/wiki/Pipes_e_filtros)
* [Plugin](https://pt.wikipedia.org/wiki/Plugin)
* [Aplicação monolítica](https://pt.wikipedia.org/wiki/Aplica%C3%A7%C3%A3o_monol%C3%ADtica)
* [Modelo em três camadas](https://pt.wikipedia.org/wiki/Modelo_em_tr%C3%AAs_camadas)
* [Analise de sistema estruturada](https://pt.wikipedia.org/w/index.php?title=Analise_de_sistema_estruturada&action=edit&redlink=1) (baseada em módulos, mas usualmente monolíticas em dentro dos módulos)
* [Arquitetura orientada a serviço](https://pt.wikipedia.org/wiki/Service-oriented_architecture)
* [Arquitetura orientada a busca](https://pt.wikipedia.org/w/index.php?title=Arquitetura_orientada_a_busca&action=edit&redlink=1)
* [MVC](https://pt.wikipedia.org/wiki/MVC)

**Estrutura**: Software;

**Elementos**: classes; componentes; serviços.

**Componentes do software => propriedades externas**

**Relação com outros softwares**

A arquitetura de software facilita a comunicação entre os stakeholders e o reuso do projeto de componentes, bibliotecas.

Mary Shaw e David Garlan abordam os conceitos em AS tais como: componentes, estilos, conexões, etc

Criação de um metamodelo que auxilia o entendimento de como o sistema irá se comportar.

Requisitos arquiteturais; Condutores arquiteturais. Visão usuário, desenvolvedor, negócio.

**Arquitetura de Sofwtare**

Arquitetura de Software

Algumas Definições:

1. **Confiabilidade**  é a capacidade de uma pessoa ou sistema de realizar e manter seu funcionamento em circunstâncias de rotina, bem como em circunstâncias hostis e inesperadas. Define a probabilidade de um sistema completar a sua função durante um intervalo de tempo.
2. **Manutenibilidade**  é um termo usado para definir que certo [Software](https://pt.wikipedia.org/wiki/Software) ou [Hardware](https://pt.wikipedia.org/wiki/Hardware) tem certa facilidade de ser mantido através de manutenções, ou seja, para algo ser manutenível deve-se existir certo cuidado ao ser produzido obedecendo a padrões. Tratando de hardware, para ele ser considerado manutenível o fabricante tem que ter peças no mercado e assistências para aplicar serviços a esse determinado produto ou pelo menos se homologado na [IEEE](https://pt.wikipedia.org/wiki/IEEE) para assim componentes de outros fabricantes sejam compatível com seu produto. No ramo de software um software tem grande manutenibilidade quando obedecer a alguns conceitos de desenvolvimento da [engenharia](https://pt.wikipedia.org/wiki/Engenharia) e [arquitetura de software](https://pt.wikipedia.org/wiki/Arquitetura_de_software). Desse modo a implementação e manutenção do [sistema](https://pt.wikipedia.org/wiki/Sistema) se torna mais rápida e precisa.
3. Alta **Disponibilidade**(HA:High-Availability) é um sistema informático resistente a falhas de [hardware](https://pt.wikipedia.org/wiki/Hardware), [software](https://pt.wikipedia.org/wiki/Software) e energia, cujo objetivo é manter os serviços disponibilizados o máximo de tempo possível.[[12]](https://pt.wikipedia.org/wiki/Sistema_de_alta_disponibilidade#cite_note-open-e.com-1)  Quanto mais [redundância](https://pt.wikipedia.org/wiki/Redund%C3%A2ncia_(inform%C3%A1tica)) existir, menores serão os [SPOF](https://pt.wikipedia.org/wiki/SPOF) (Single Point Of Failure), e menor será a probabilidade de interrupções no serviço.
4. **Segurança** é a percepção de se estar protegido de riscos, perigos ou perdas.
5. **Portabilidade**  é a sua capacidade de ser [compilado](https://pt.wikipedia.org/wiki/Compilado) ou executado em diferentes [arquiteturas](https://pt.wikipedia.org/wiki/Arquitetura_de_computador) (seja de [hardware](https://pt.wikipedia.org/wiki/Hardware) ou de [software](https://pt.wikipedia.org/wiki/Software)). O termo pode ser usado também para se referir a re-escrita de um [código fonte](https://pt.wikipedia.org/wiki/C%C3%B3digo_fonte) para uma outra [linguagem de computador](https://pt.wikipedia.org/wiki/Linguagem_de_computador).
6. **Escalabilidade** é uma característica desejável em todo o sistema, em uma rede ou em um processo, que indica sua habilidade de manipular uma porção crescente de trabalho de forma uniforme, ou estar preparado para crescer.
7. **Desempenho** seria o tempo de processamento de uma requisição externa. **Responsividade** tempo de reação a uma requisição. **Latência** tempo mínimo para qualquer requisição, mesmo nulo. **Thronghput** quando pode ser feito em um dado intervalo de tempo. **Carga** stress suportado por um sistema, geralmente medido em número de usuários. **Sensibilidade a carga** variação do tempo de resposta em relação a carga. **Eficiência** desempenho dividido pelos recursos.

Definir a arquitetura em camadas (Decisões de Arquitetura = Custo).

1. Interface caixa preta
2. Interface caixa branca
3. Domínio: Módulos do Projeto
4. Serviços: SGBDR; Visões; SGBDOO
5. Plataforma: Linguagem; SO
6. Servidor e Estações de Trabalho
7. Distribuição Geográfica

Ex1.: Sistema Desktop Local (Uma das camadas da arquitetura de software)

Camada Monolítica: Interface; Regra de Negócio; DAO (Acesso a dado); Servidor de BD Local; Mono Usuário

1 Computador

Sistema Local

SGBD

Ex2.: Sistema Desktop Distribuído - Rede (Uma das camadas da arquitetura de software); Vários usuários

2 Camadas: (Interface; Regra de Negócio; DAO (Acesso a dado)); Servidor de BD Distribuído.

N Computadores

SGBD

Várias Aplicações Desktop

Ex3.: Sistema Web - (Uma das camadas da arquitetura de software); Conexões TCP/IP

3 Camadas: 3.1 Camada de Apresentação; 3.2 Camada de Aplicação; 3.3 Camada BD; Servidor BD Distribuído

TCP/IP

POOL DAO

Camada de BD

Camada de Aplicação/Server

Camada de Apresentação

SGBDR

Ex4.: Sistema Web - (Uma das camadas da arquitetura de software); Conexões TCP/IP

3 Camadas: 3.1 Camada de Apresentação; 3.2 Camada Web; 3.3 Camada de Aplicação; 3.4 Camada BD; Servidor BD Distribuído;

Na camada de apresentação: Celular; Smart Card; Console APP; Browser; PDA.

Camada de Apresentação

N Camadas WEB

N Camadas de Aplicação/Server

TCP/HTTP

Camada de BD

SGBDR

SGBDR

N

Referências

1. [Ir para cima↑](https://pt.wikipedia.org/wiki/Mary_Shaw#cite_ref-1) [Bass, Len](https://pt.wikipedia.org/w/index.php?title=Len_Bass&action=edit&redlink=1" \o "Len Bass (página não existe)). Software architecture in practice. Pearson Education India, 2007.
2. [Ir para cima↑](https://pt.wikipedia.org/wiki/Mary_Shaw#cite_ref-RTF_2000_2-0) Fielding, Roy Thomas. [Architectural styles and the design of network-based software architectures](http://jpkc.fudan.edu.cn/picture/article/216/35/4b/22598d594e3d93239700ce79bce1/7ed3ec2a-03c2-49cb-8bf8-5a90ea42f523.pdf). Diss. University of California, Irvine, 2000.
3. ↑ [Ir para:a](https://pt.wikipedia.org/wiki/Mary_Shaw#cite_ref-IEEE_2013_3-0) [b](https://pt.wikipedia.org/wiki/Mary_Shaw#cite_ref-IEEE_2013_3-1) [Mary Shaw: Biography](http://www.ieeeghn.org/wiki/index.php/Mary_Shaw), ieeeghn.org/wiki, 2013. Acessado em 13 de setembro de 2015.
4. [Ir para cima↑](https://pt.wikipedia.org/wiki/Mary_Shaw#cite_ref-CMU_2011_4-0) [ACM SIGSOFT, Outstanding Research Award winners](http://www.sigsoft.org/awards/outResAwd.htm), sigsoft.org. Acessado em 13 de setembro de 2015.
5. [Ir para cima↑](https://pt.wikipedia.org/wiki/Mary_Shaw#cite_ref-5) Carnegie Mellon University, "[Carnegie Mellon's Shaw and Garlan Honored For Pioneering Research in Software Architecture](http://www.cmu.edu/news/archive/2011/May/may2_shawgarlanaward.shtml)," cmu.edu/news. Press release, May 2, 2011. Acessado em 13 de setembro de 2015.
6. [Ir para cima↑](https://pt.wikipedia.org/wiki/Mary_Shaw#cite_ref-6) The White House, Office of the Press Secretary,[President Obama Honors Nation's Top Scientists and Innovators](http://www.whitehouse.gov/the-press-office/2014/10/03/president-obama-honors-nation-s-top-scientists-and-innovators), 3 de outubro de 2014.
7. [Ir para cima↑](https://pt.wikipedia.org/wiki/Mary_Shaw#cite_ref-7) [Lista de publicações](http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/s/Shaw:Mary.html) do [Digital Bibliography & Library Project](https://pt.wikipedia.org/wiki/Digital_Bibliography_%26_Library_Project).
8. [Ir para cima↑](https://pt.wikipedia.org/wiki/Mary_Shaw#cite_ref-SG_1996_8-0) Shaw & Garlan (1996, summary)
9. M. Shaw, D. Garlan; Software Architecture. Perspectives on an Emerging Discipline, Prentice Hall, 1996.
10. D. Garlan and Mary Shaw. An introduction to software architecture. Technical Report- CMU-CS-94166,Carnegie Mellon University, January 1994.
11. Len Bass, Paul Clements, and Rick Kazman. Software Architecture in Practice. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998., 2nd edition, April 2003.
12. [HA Storage Cluster](http://www.open-e.com/site_media/download/documents/ha-storage-cluster.pdf) (em inglês) *open-e.com*. Visitado em 07/11/2015.

Outros

1. http://pangeanet.org/