Arquitetura de Software

Prof. Jacson Barbosa
Curso: Bacharelado em Engenharia de Software

Aula 02
Introdução à Arquitetura de Software

Referência Bibliográfica

- *Alguns conteúdos apresentados possuem adaptações das aulas do*
  - *Prof. Dr. Valdemar V. Graciano Neto*

Roteiro

- Introdução
- Definições de Arquitetura de Software
- Conceitos
  - Padrão Arquitetural
  - Visão e Ponto de Vista
  - Decisão Arquitetural
  - Outros
- Proposta de Atividade

Introdução

- Software é um produto de Engenharia;
- Entretanto, software é um produto abstrato e não tangível;
- Consequentemente, visualizar o "formato" do software é uma atividade complexa;
  - Adicionalmente, mesmo em outras engenharias, não se tem o formato do produto até que ele esteja construído, a não ser pelos modelos (maquete, etc.)

Introdução

- Paralelamente, é difícil construir qualquer coisa cujo formato não seja conhecido;
- Logo, a definição de arquitetura para um produto de engenharia é uma atividade essencial para garantir a qualidade do produto;

Introdução

- Na Arquitetura Civil, arquitetura diz respeito à "forma" das construções:
  - Neoclássica;
  - Barroca;
  - Gótica; etc.

Fonte:
http://thaa2.files.wordpress.com/2009/07/arquitetura-neoclassica-2.jpg

Introdução

Fonte:
http://oglobo.globo.com/blogs/arquivos_upload/2009/01/129_302-Francsico,%20Ouro%20Preto.jpg

Fonte:
http://3.bp.blogspot.com/-_hkObXOms_o/T6sBYOwUBDI/AAAAAAAAAEk/d5F3oxSqDhs/s1600/4390-g.jpg

Definições

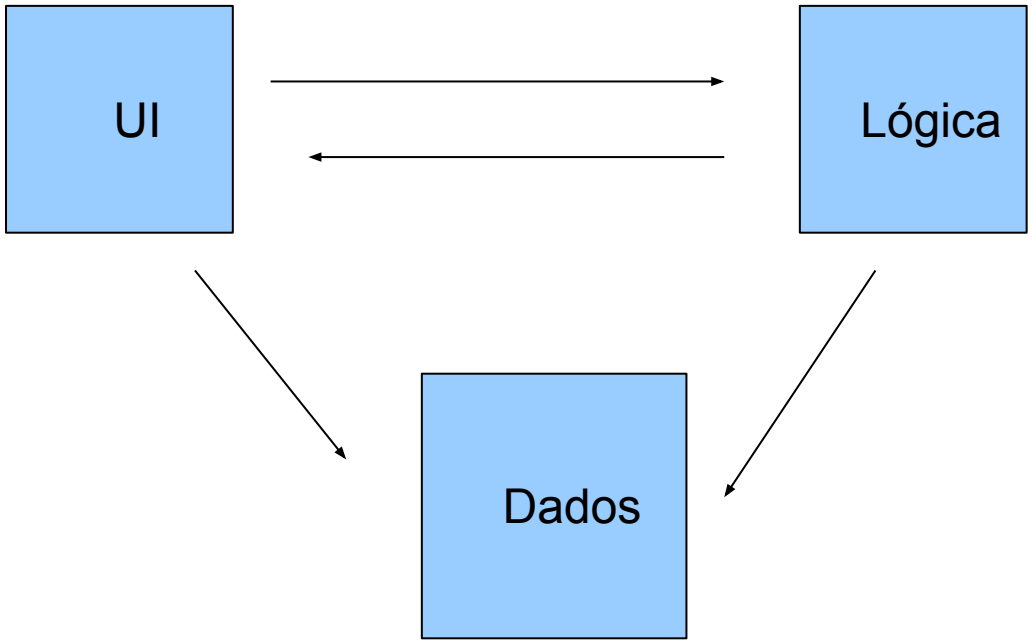
- Afinal, o que é uma Arquitetura de Software?

Definições

- Afinal, o que é uma Arquitetura de Software?
  - Primeira coisa a saber: não há definição consensual;
  - Informal:
    - Diz respeito à forma como porções de código são agrupadas em partes específicas, e como essas partes interagem entre si para culminar no funcionamento do software.

Definições

- Afinal, o que é uma Arquitetura de Software?
  - Informalmente:



Definições

- Martin Fowler:

design

Editor: Martin Fowler ■ ThoughtWorks ■ fowler@acm.org

Who Needs an Architect?

Martin Fowler

Wandering down our corridor a while ago, I saw my colleague Dave Rice in a particularly grumpy mood. My brief question caused a violent statement, "We shouldn't interview anyone who has 'architect' on his resume." At first blush, this was an odd turn of phrase, because we usually introduce Dave as one of our leading architects.

The reason for his title schizophrenia is the fact that, even by our industry's standards, "architect" and "architecture" are terribly overloaded words. For many, the term "software architect" fits perfectly with the smug controlling image at the end of *Matrix Reloaded*. Yet even in firms that have the greatest contempt for that image, there's a vital role for the technical leadership that an architect such as Dave plays.

What is architecture?

When I was fretting over the title for *Patterns of Enterprise Application Architecture* (Addison-Wesley, 2002), everyone who reviewed it agreed that "architecture" belonged in the title. Yet we all felt uncomfortable defining the word. Because it was my book, I felt compelled to take a stab at defining it.

My first move was to avoid fuzziness by just letting my cynicism hang right out. In a sense, I define *architecture* as a word we use when we want to talk about design but want to puff it up to make it sound important. (Yes, you can imagine a similar phenomenon for architect.) However, as so often occurs, inside the blighted cynicism is a pinch of truth. Understanding came to me after reading a posting from Ralph Johnson on the Extreme Programming mailing list. It's so good I'll quote it all.

A previous posting said

> The RUP, working off the IEEE definition, defines architecture as "the highest level concept of a system in its environment. The architecture of a software system (at a given point in time) is its organization or structure of significant components interacting through interfaces, those components being composed of successively smaller components and interfaces."

Johnson responded:

> I was a reviewer on the IEEE standard that used that, and I argued uselessly that this was clearly a completely bogus definition. There is no highest level concept of a system. Customers have a different concept than developers. Customers do not care at all about the structure of significant components. So, perhaps an architecture is the highest level concept that developers have of a system in its environment. Let's forget the developers who just understand their little piece. Architecture is the highest level concept of the expert developers. What makes a component significant? It is significant because the expert developers say so.
>
> So, a better definition would be "In most successful software projects, the expert developers working on that project have a shared understanding of the

2 IEEE SOFTWARE Published by the IEEE Computer Society 0740-7459/03/$17.00 © 2003 IEEE

Fonte:
http://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf
Acesso: Agosto de 2014

Definições

- Fowler
  - "O conceito de mais alto nível que os desenvolvedores têm de um sistema [...] em seu ambiente";
  - "Uma consciência compartilhada do projeto (*design)* do sistema [...] e inclui como o sistema é dividido em componentes e como estes componentes interagem através de interfaces."
  - "O conjunto de decisões de projeto que devem ser feitas de forma precoce em um projeto".

Definições

- Clements
  - "O conjunto de estruturas necessárias para compreender o sistema, e compreender os elementos, relações entre eles, e propriedades de ambos."
  - "Software architecture is composed of elements, connections or relations among them, and, usually, some other aspect or aspects, such as configuration; constraints or semantics; analyses or properties; or rationale, requirements, or stakeholders' needs. "

Definições

- Mary Shaw e David Garlan
  - A descrição de elementos a partir dos quais sistemas são construídos, interações entre estes elementos, padrões que guiam sua composição, e restrições sobre estes padrões.
  - Elementos (componentes) neste contexto são clientes e servidores, banco de dados, filtros, camadas, etc.
  - Interações podem ser: invocação de procedimentos ou acesso compartilhado a variáveis; ou, protocolos de cliente-servidor, de acesso a BD, fluxos, multicast assíncrono, etc.

Definições

- SWEBOK V3
  - O conjunto de estruturas necessário para compreender o sistema, os elementos de software, a relação entre eles, e as propriedades de ambos.

Definições

- IEEE 1471-2000
  - The fundamental organization of a system embodied in its components, their relations to each other, and to the environment, and the principles guiding its design and evolution. (IEEE 1471 2000, p. 9)

Definições

- Dicionário Computação (IEEE 610)
  - A estrutura organizacional de um sistema ou componente. (Genérico para arquitetura).

Definições

- Resumindo:
  - Clements et. Al: "“architecture is what you get before you start adding detail to the design."
  - Arquitetura de Software diz respeito ao "formato" do software, às partes principais do software, como interagem, suas interfaces, e todos os elementos envolvidos no funcionamento e ambientação do software.

Vantagens em Definir uma Arquitetura de Software

- Manutenibilidade;
- Reutilização;
- Encapsulamento;
- Rastreabilidade;
- Qualidade;
- Estabelece vocabulário para diálogo;

1 . Exercício Rápido

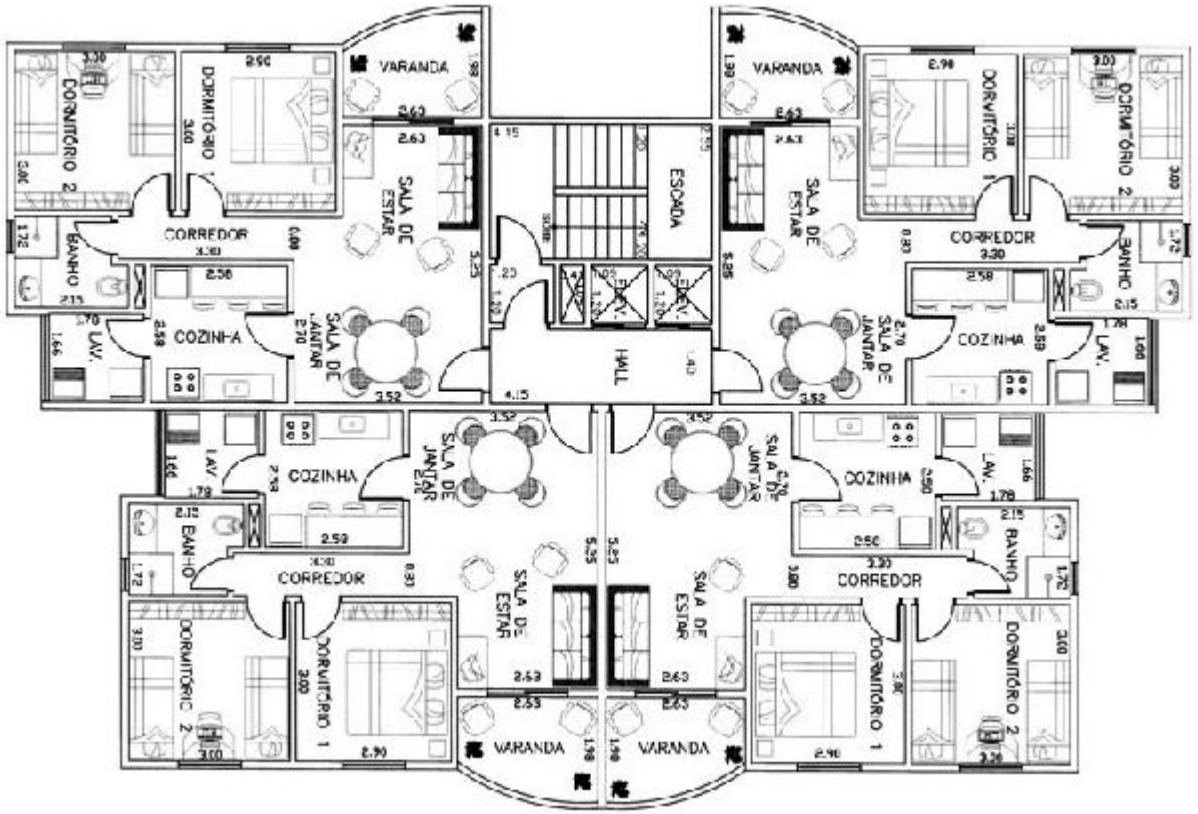
- Construa sua própria definição de Arquitetura de Software de acordo com as definições que foram vistas e conceitos apreendidos.

Conceitos

- Visão e Pontos de Vista;
- Decisão Arquitetural;
- Padrão Arquitetural;
- Concerns;
- Atributos de Qualidade;
- Linguagem de Descrição Arquitetural;
- Framework Arquitetural;
- Framework de Aplicação;
- Reutilização.

Visões

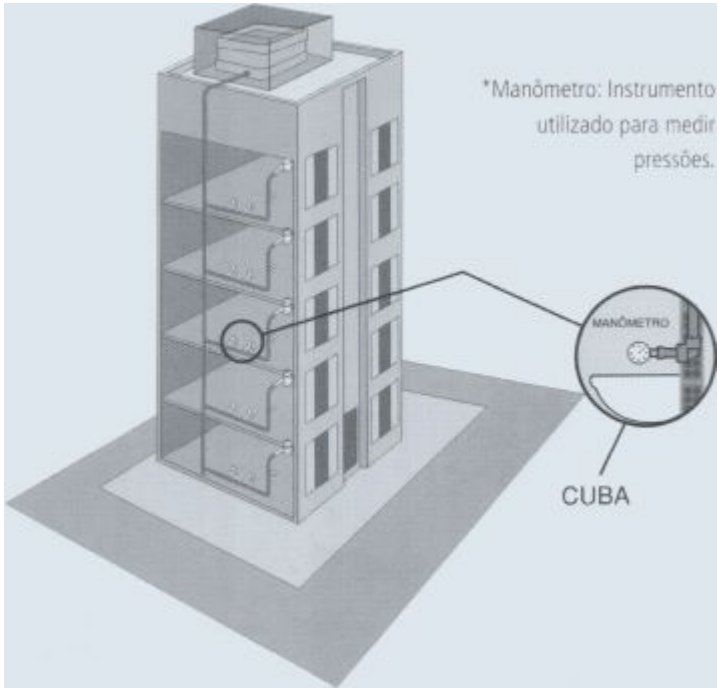
- Analogia com Prédio (Visão da Planta)



Fonte: http://www.conexaolivre.com.br/arquitetura/projetos.grupo2/slides/34-%20Planta%20Edificio%20Sao%20Mauro.jpg

Visões

- Analogia com Prédio (Visão Hidráulica)



Fonte: http://www.renatomassano.com.br/dicas/residencial/conceitos_fundamentais/sistema_predial_02.jpg

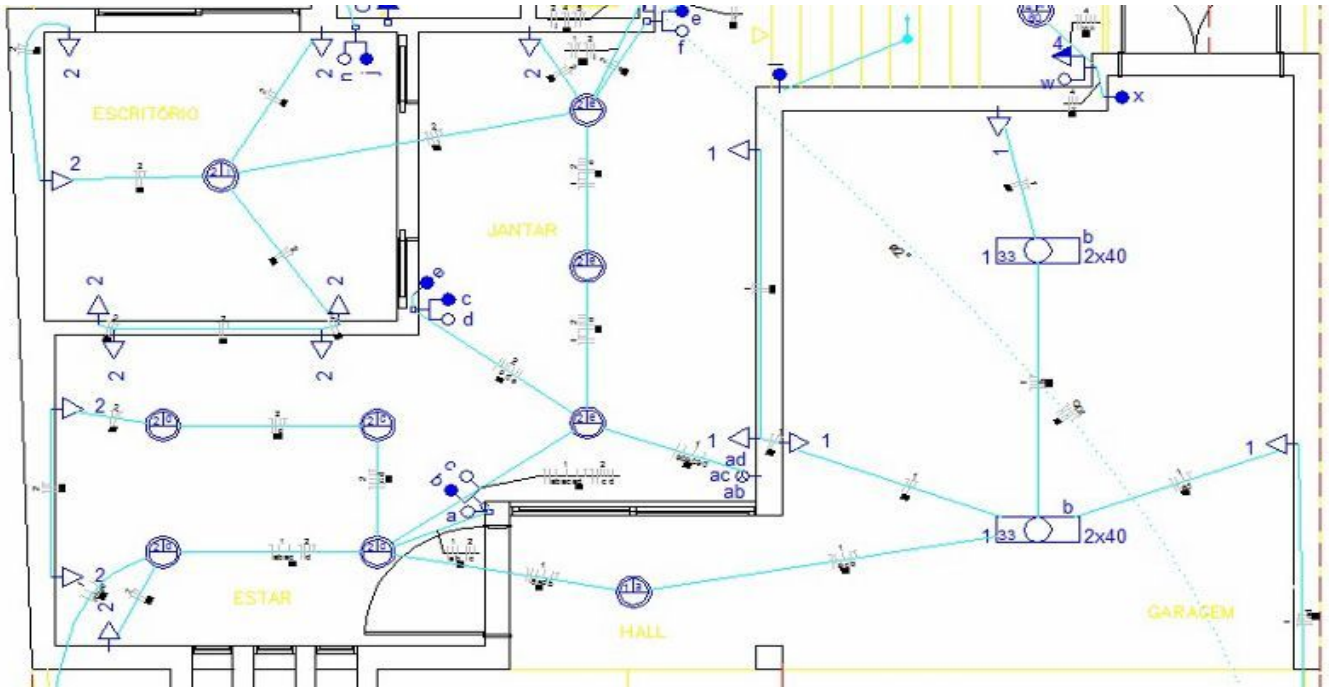
Visões

- Analogia com Prédio (Visão Externa)

Fonte: http://1.bp.blogspot.com/-3iQRgm_3-bM/UiSqkA6RPpI/AAAAAAAACYI/dy7brmIrIGQ/s1600/maquete_predio_acrop_.jpg

Visões

- Analogia com Prédio (Visão do Projeto Elétrico)



Fonte: http://www.andreiafigueiredo.com.br/portfolio/Servicos/slides/ELETRICO.jpg

Visões

- Todas as visões juntas formam a arquitetura do prédio.
- Sozinha, nenhuma delas é capaz de representar o prédio por inteiro.
- Mas juntas, elas formam toda a estrutura do prédio.

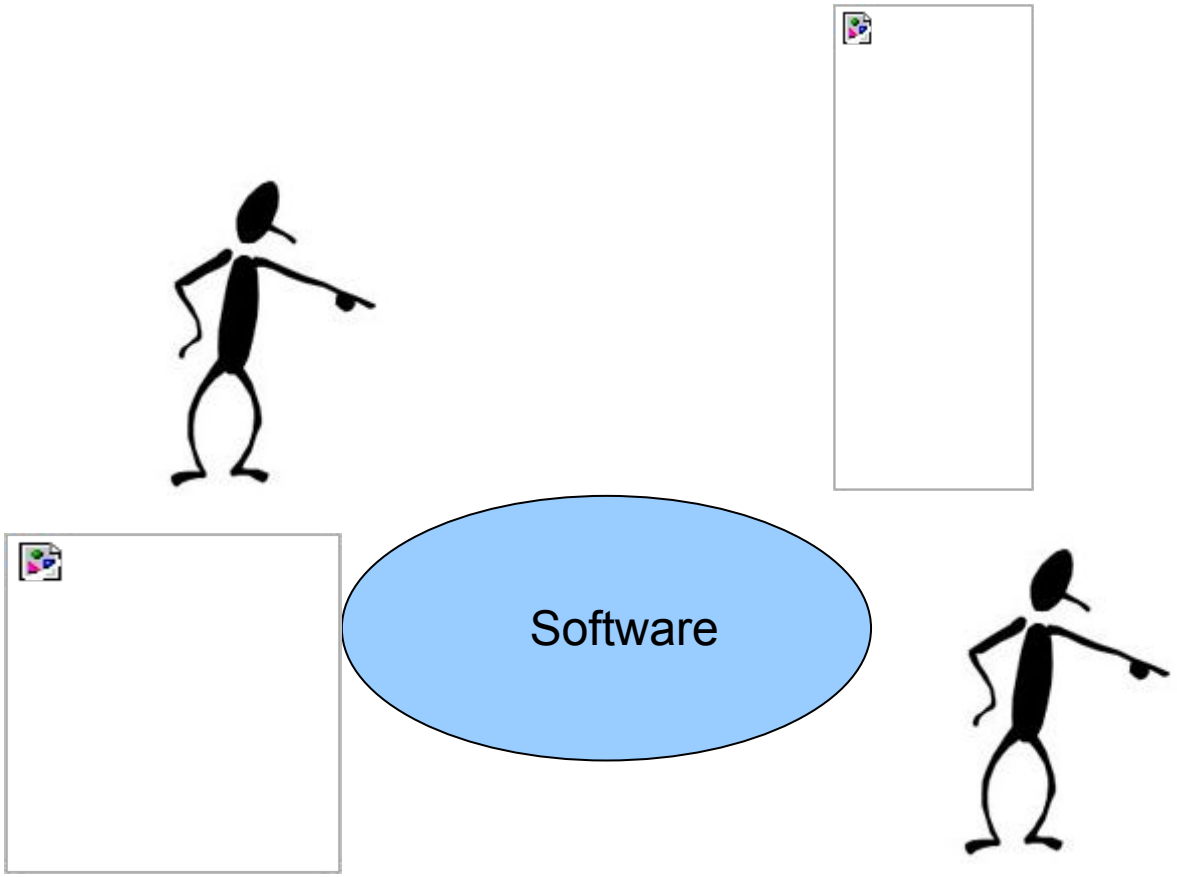
Visões

- Em software, existem também vários modelos que representam várias visões do software como:
  - Visão de Estrutura (principais componentes, classes, etc.);
  - Visão de Comportamento (interação entre elementos, fluxo de informações, etc.);
  - Dentre outros.
- O importante é ser o mais exaustivo possível de modo a representar o software em sua integridade.

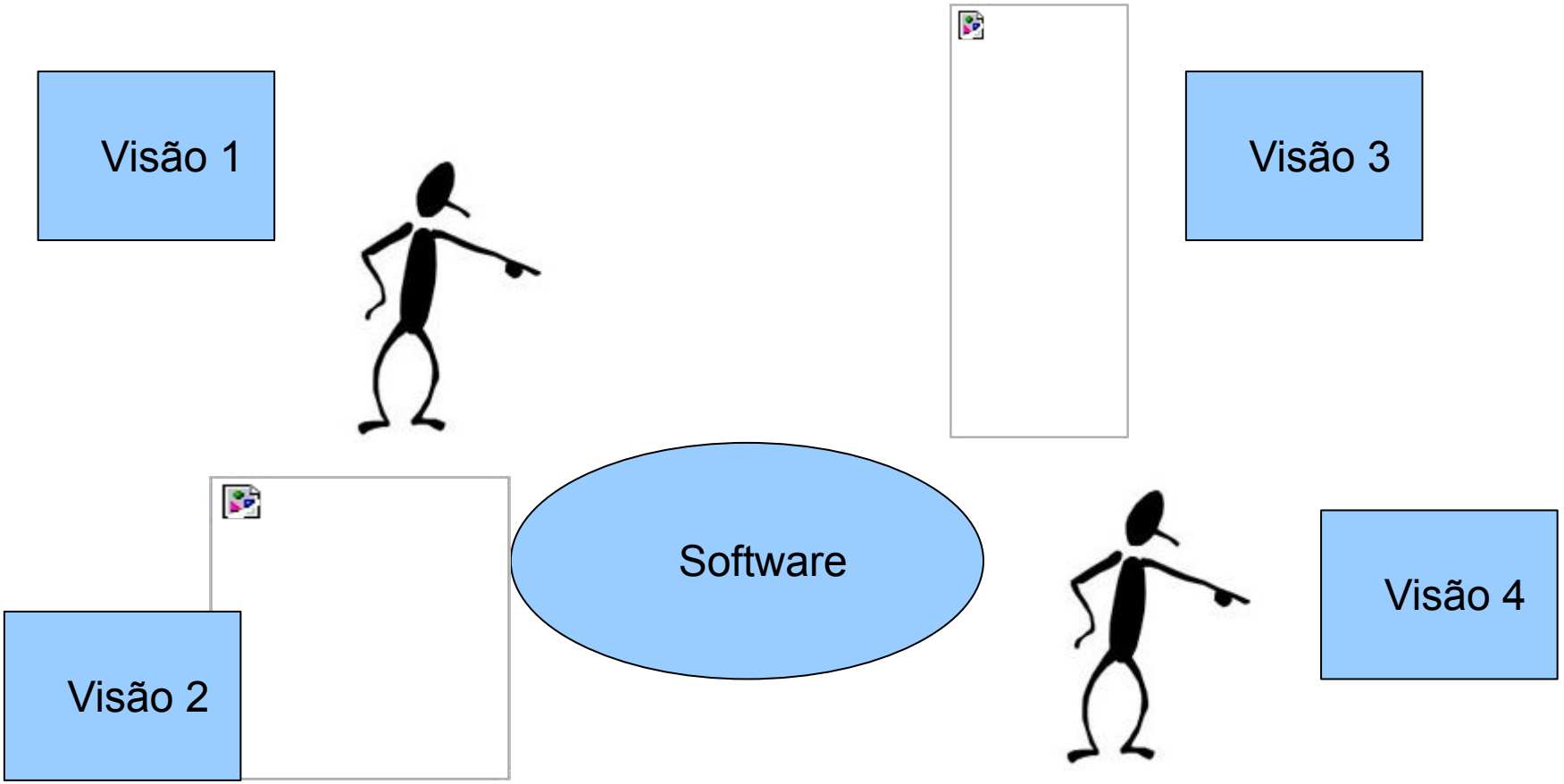
Ponto de Vista

- Um ponto de vista origina uma visão.
- O engenheiro eletricista vê o prédio sob a ótica de projeto elétrico; o engenheiro hidráulico sob a ótica de hidráulica, e por aí vai. E cada um gera um modelo diferente.
- Papéis diferentes olhando para um mesmo objeto geram modelos diferentes sobre diferentes aspectos do mesmo objeto observado (ponto de vista gera visão)

Ponto de Vista



Ponto de Vista



Visões

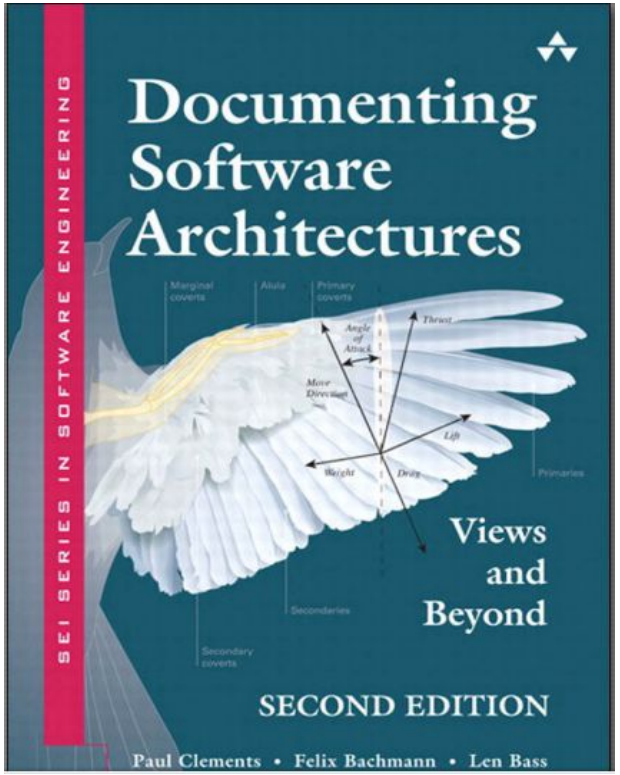
- Sugestões da Literatura:
  - 4+1 Views
  - Views and Beyond (Clements)

2. Exercício Rápido

- Quantas e quais visões acredita ser suficiente para representar adequadamente a arquitetura de um software?
- Depende de domínio?

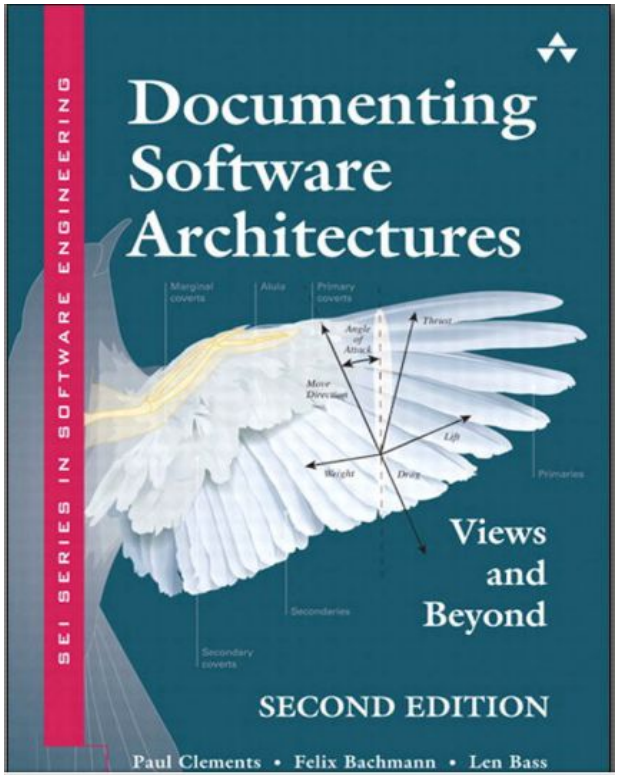
Decisão Arquitetural

- Decisões que permitem ao sistema contemplar seus atributos de qualidade e outros requisitos.
- Ex.:
  - Qual o SGBD para garantir tempo de resposta em *x* ms?
  - Qual organização de componentes para garantir segurança, ou tolerância a falhas (sistemas críticos)?



Decisão Arquitetural

- Como registrar estas decisões arquiteturais?
- Uma possibilidade:
  - Informal;
  - Formal;
  - Ontologias (Clements et. al)



Padrão Arquitetural

- Padrão (Erich Gamma):
  - Uma solução sistemática, repetível, e adaptável para um problema recorrente.
  - Envolve (uma trinca + o nome):
    - Problema recorrente;
    - Contexto;
    - Solução padronizada.

Padrão Arquitetural

- Exemplo:
  - Padrão (de Projeto) Singleton:
    - Problema: Criar um objeto único dentro de um sistema (exemplo: objeto representando uma conta bancária);
    - Contexto: Banco;
    - Solução: tornar o método de criação estático e final; construtor *private*; e invocar criação com construtor de dentro do método.

Padrão Arquitetural

- Problema, Contexto, e Solução para criação de arquiteturas de software;
- Exemplo: Camadas, Componentes, SOA, Cliente-Servidor, etc.
- Também chamados de Estilos Arquiteturais.

Conceitos

- Visão e Pontos de Vista
- Decisão Arquitetural
- Padrão Arquitetural
- Concerns
- Atributos de Qualidade
- Linguagem de Descrição Arquitetural
- Framework Arquitetural
- Framework de Aplicação
- Reúso

Concerns

- *Concerns* = Interesses;
- Separação de Interesses é uma grande recomendação de design de sistemas;
- Um *concern* é uma parte do problema que queremos tratar como uma unidade conceitual única na solução do software.
- Definir uma arquitetura de software está inerentemente ligada à tarefa de separação de *concerns*.
- A Separação de *concerns* tenta resolver as limitações da cognição humana ao lidar com a complexidade do software.

Fonte: http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0016030_04_cap_02.pdf

Concerns

- Os sistemas de software complexos devem ser decompostos em unidades modulares menores e claramente separadas, cada uma lidando com um único *concern*.
- Exemplos de *concern*:
  - Persistência;
  - Concorrência;
  - Interação Humano-Computador;
  - Lógica;
  - Negócio;

Fonte: http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0016030_04_cap_02.pdf

Concerns

- Orientação a Aspectos é um exemplo de uma técnica utilizado para encapsulamento de interesses (*concerns*) chamados de transversais;
- Transversais = espalhados no código, repetidos;
- Exemplo:
  - Concorrência;
  - Logging;
  - Tratamento de Exceções;
  - Persistência;

Fonte: http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0016030_04_cap_02.pdf

Concerns

- Isolar concerns é um trabalho inerente à definição de Arquitetura de Software.

Conceitos

- Visão e Pontos de Vista
- Decisão Arquitetural
- Padrão Arquitetural
- Concerns
- Atributos de Qualidade
- Linguagem de Descrição Arquitetural
- Framework Arquitetural
- Framework de Aplicação
- Reutilização

Atributos de Qualidade

- Requisitos Não-Funcionais que são utilizados para aferir a qualidade de um sistema;
- Exemplos:
  - Desempenho
  - Disponibilidade
  - Modificabilidade
  - Segurança
  - Testabilidade
  - Usabilidade

Conceitos

- Visão e Pontos de Vista
- Decisão Arquitetural
- Padrão Arquitetural
- Concerns
- Atributos de Qualidade
- Linguagem de Descrição Arquitetural
- Framework Arquitetural
- Framework de Aplicação
- Reutilização

Atributos de Qualidade

- Em geral, a definição de uma arquitetura auxilia o engenheiro no cumprimento de atributos de qualidade.
- Exemplo:
  - Tolerância a Falhas pode ser conseguida através de Redundância de unidades de processamento, o que faz parte da definição da arquitetura;
  - Escolha de tecnologias influenciam no desempenho;
  - Disponibilidade está ligada a Redundância também;
  - Auto-Adaptação em SMSS está ligado a um loop de adaptação, que faz parte da definição da arquitetura;
  - Estas estratégias são chamadas de **táticas arquiteturais.**

Linguagem de Descrição Arquitetural

- ADL: Architectural Description Language;
- Uma linguagem utilizada para especificar uma arquitetura;
- Em geral em termos de componentes, conectores, e comportamento;
- Exemplo: Pi-ADL, ACME, etc.

Conceitos

- Visão e Pontos de Vista
- Decisão Arquitetural
- Padrão Arquitetural
- Concerns
- Atributos de Qualidade
- Linguagem de Descrição Arquitetural
- Framework Arquitetural
- Framework de Aplicação
- Reutilização

Framework Arquitetural

- Conjunto de recomendações para criação de uma arquitetura (de software, e corporativa);
- Exemplos:
  - TOGAF;
  - DODAF;
  - Zachman;

Conceitos

- Visão e Pontos de Vista
- Decisão Arquitetural
- Padrão Arquitetural
- Concerns
- Atributos de Qualidade
- Linguagem de Descrição Arquitetural
- Framework Arquitetural
- Framework de Aplicação
- Reutilização

Framework de Aplicação

- Um Framework de Aplicação (FA) é uma aplicação semi-completa, construída como uma coleção organizada de componentes de software reusáveis para facilitar a implementação de aplicações de software customizadas.
- Um conjunto de classes que constitui um esqueleto de um produto de software: ele contém lacunas (ou *hot-spots*) que devem ser preenchidas usando código manualmente inserido e específico para o produto.

Framework de Aplicação

- As principais características de um FA são modularidade, reúso, extensibilidade, e Inversão de Controle (IC).
- Frameworks de Aplicação têm se tornado o padrão *de-facto* para implementar sistemas de negócio
- Exemplos:
  - Swing; JSF; Spring; JPA;

Framework de Aplicação

- As aplicações desenvolvidas com um FA são necessariamente **aderentes à arquitetura do FA.**
- Logo, FA representa uma técnica de reúso arquitetural;

Conceitos

- Visão e Pontos de Vista
- Decisão Arquitetural
- Padrão Arquitetural
- Concerns
- Atributos de Qualidade
- Linguagem de Descrição Arquitetural
- Framework Arquitetural
- Framework de Aplicação
- Reutilização

Reutilização

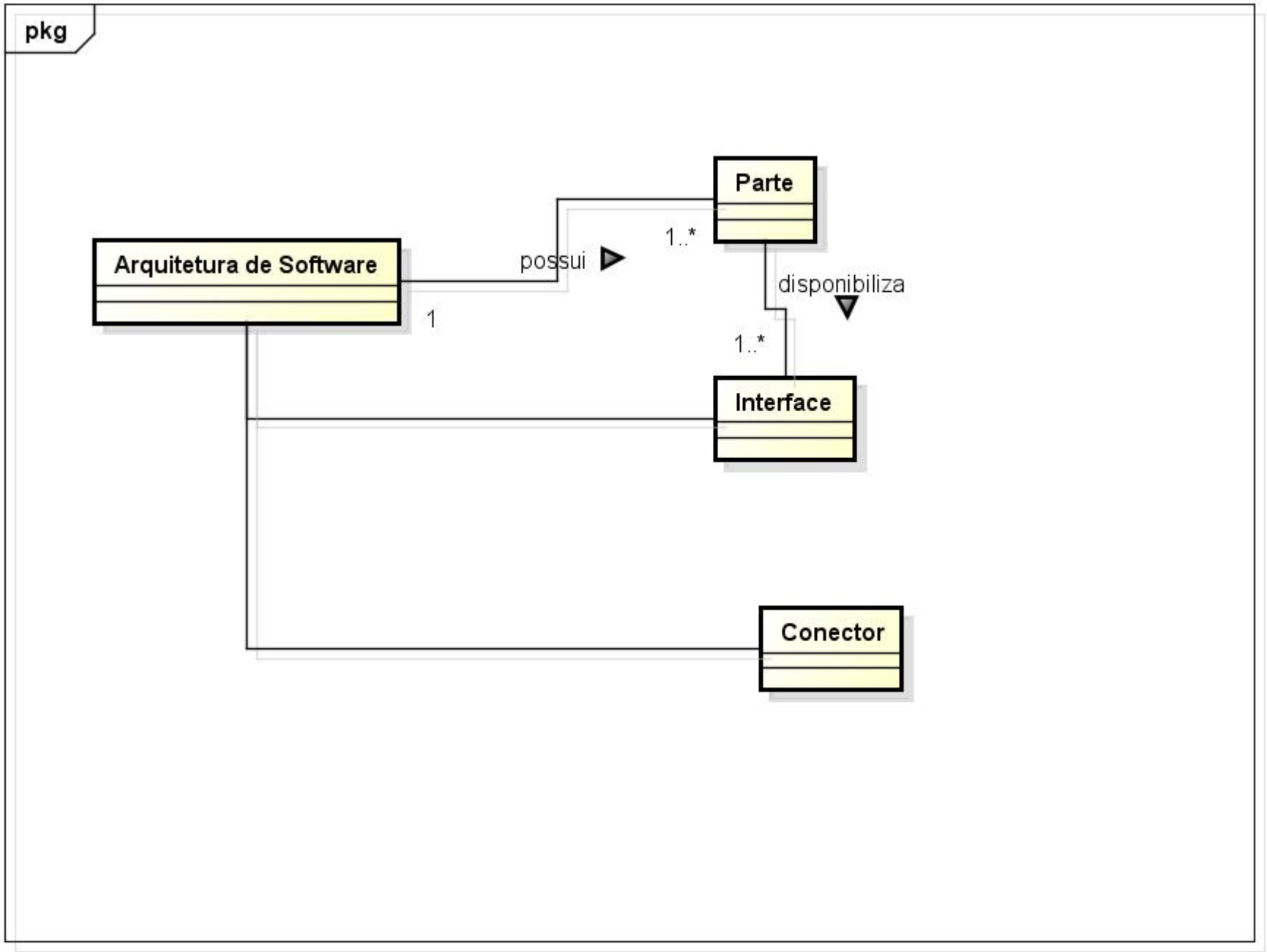
- A regra de ouro da Engenharia de Software para garantir produtividade e qualidade de software;
- Consiste no reaproveitamento de componentes, código, documentos, e demais artefatos de engenharia, para desenvolver um novo produto derivado de um anterior;
- Exemplos:
  - Linhas de Produto;
  - FA;
  - Desenvolvimento de Software Baseado em Componentes;

Reutilização

- O uso de arquitetura de software já fomenta, se bem realizada, a reutilização de software.

Proposta de Exercício

- 1. Construa um modelo conceitual utilizando **Diagrama de Classes da UML**, representando todos os **conceitos de arquitetura de software vistos em sala e estudados extra-classe**.
  O produto deve ser similar a um mapa mental, mas em **UML**.
  Algo como o que consta no slide seguinte;
- Dois alunos serão sorteados para apresentar seu diagrama na próxima aula.


pkg
Parte
1..*
Arquitetura de Software
possui
disponibiliza
1
1..*
Interface
Conector
powered by Astah