Princípios de Arquitetura de software

Bruna Diirr

brunadiirr@ic.uff.br

Divisão da fase de Projeto

Projeto geral ou preliminar

Tradução da especificação do sistema em termos da arquitetura a partir do conhecimento adquirido com requisitos

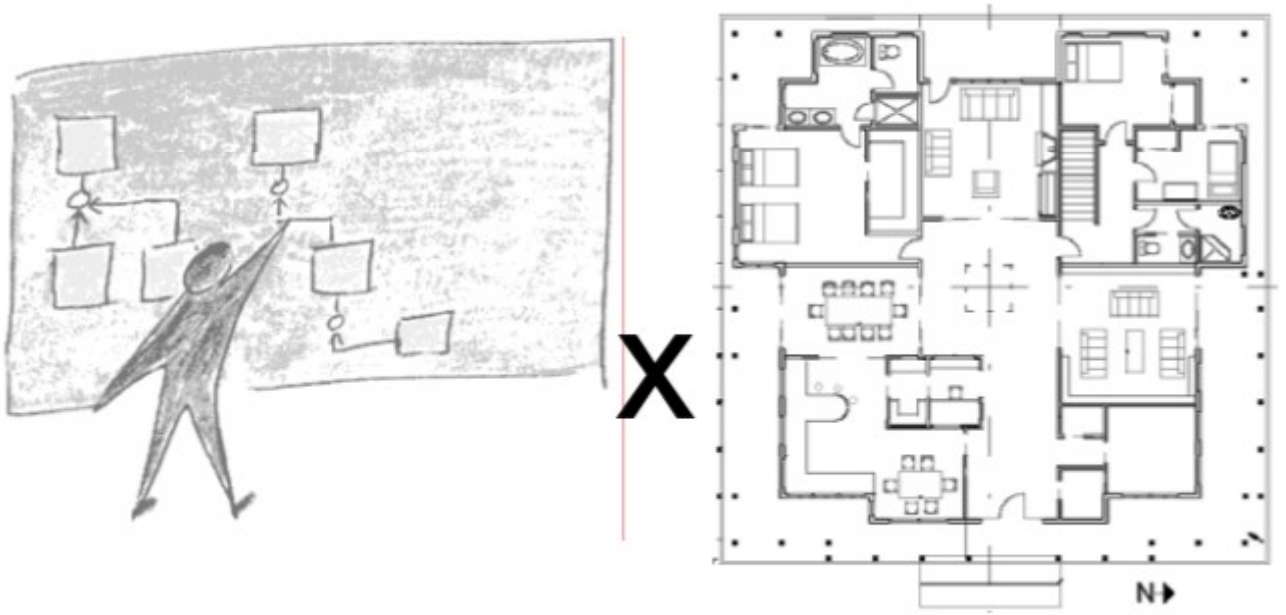
- Descrição da organização fundamental do sistema através de seus diversos módulos para alcançar os objetivos propostos pelo cliente → *Projeto da arquitetura do software*

Projeto detalhado

Refinamento progressivo e adição de detalhes (cada módulo) à arquitetura visando a codificação

O que é arquitetura de software?

Sistemas de software x Sistemas físicos



O que é arquitetura de software?

Todos os sistemas físicos maduros (prédios, aviões, automóveis, navios, pontes etc.) apresentam uma arquitetura estável

- Arquitetura
  - "Arte ou técnica de projetar uma edificação ou um ambiente de uma construção"
  - *Precede a etapa de construção da obra*
  - *Determina as partes de uma construção e como estas devem interagir*
  - *Garante a unidade da obra, ou seja, a consistência entre as suas partes*

Essas arquiteturas evoluíram ao longo do tempo

- Tentativa e erro
- Reuso e refinamento de soluções de sucesso
- Avaliação quantitativa com métodos analíticos

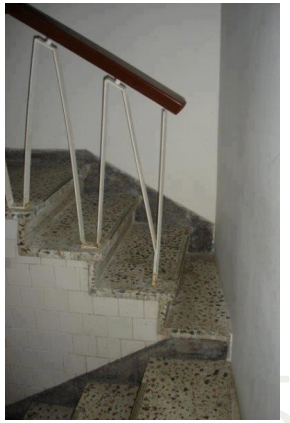
O que é arquitetura de software?

Então, como desenvolver sistemas mais complexos sem pensar em arquitetura? Como fica a qualidade desses sistemas?





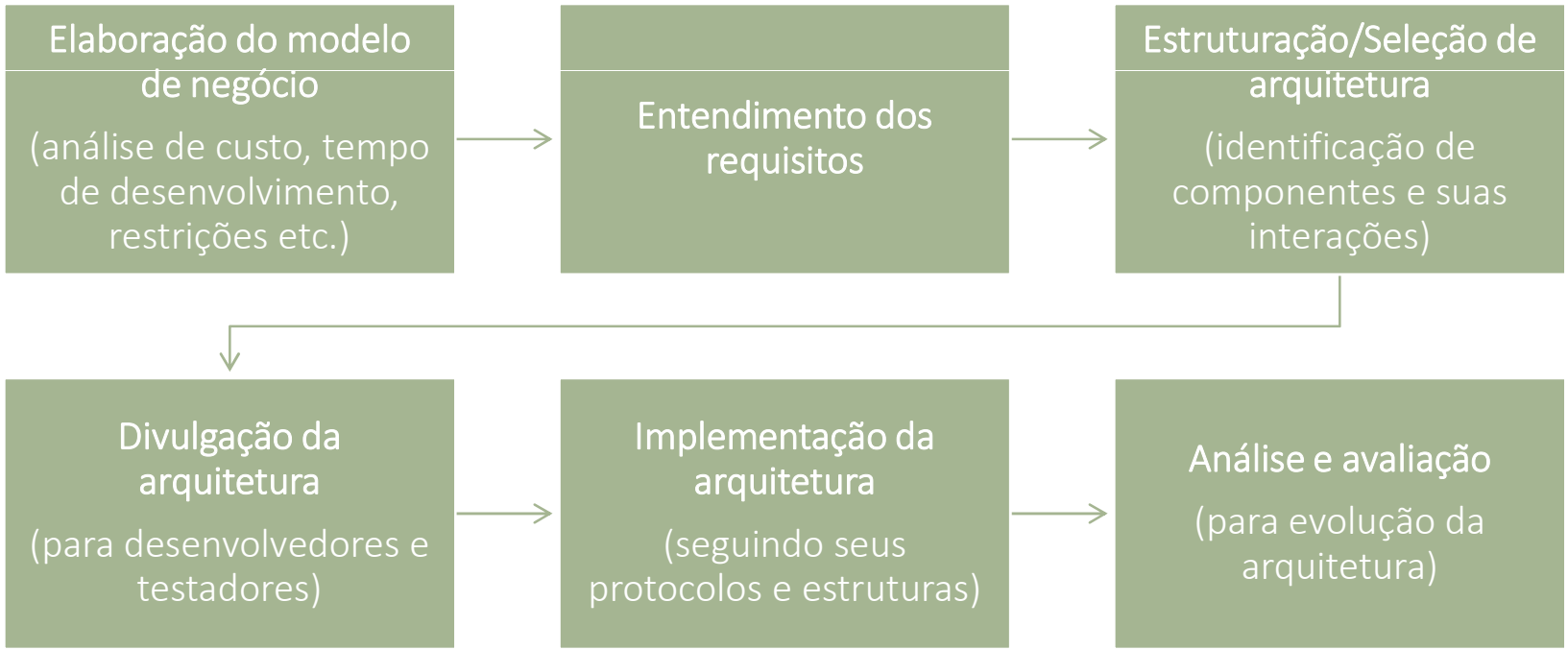




O que é arquitetura de software?

Designa processo e produto

Processo: Área da Engenharia de Software que detalha estruturas de software, visando reduzir complexidade



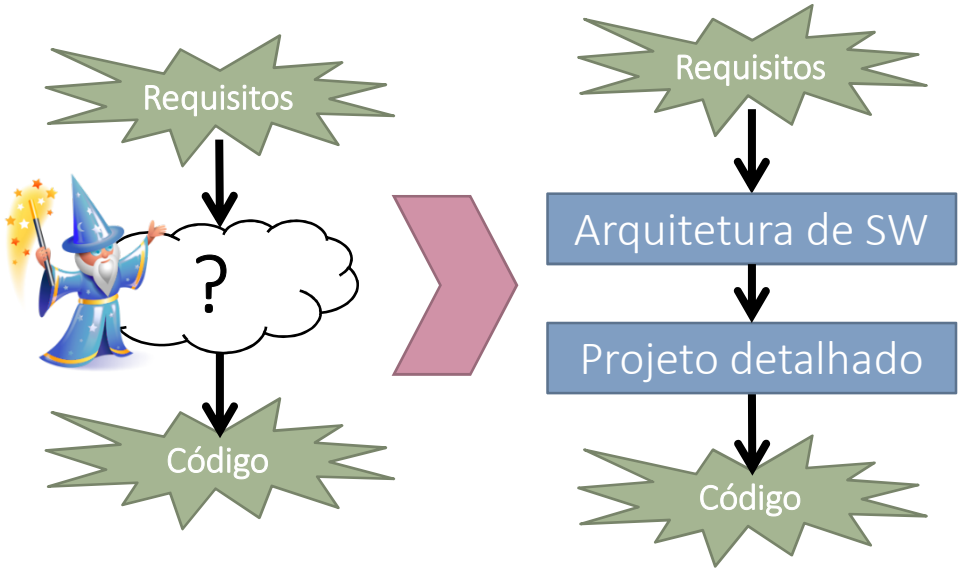
O que é arquitetura de software?

Designa processo e produto

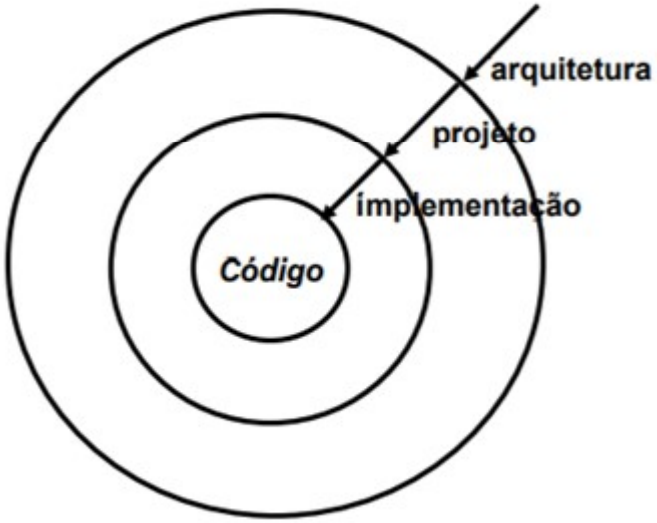
- Produto: Representação da estrutura de software
  - Descrição em alto nível de abstração que permite uma visão completa dos elementos do sistema
    - Módulos, componentes, camadas, subsistemas, classes
  - Representação das interfaces que conectam esses elementos
    - Interação entre eles ocorrem através das interfaces
  - Omite informação sobre detalhes de implementação
  - Comportamento dinâmico do sistema levado em conta
    - Fluxo de informação entre elementos, execução paralela ou sequencial de programas, efeitos em dados etc... → Estrutura dinâmica definida em tempo de execução

O que é arquitetura de software?

Papel no desenvolvimento



Ligação entre especificação e código (passando pelo projeto detalhado)



Restringe projeto e implementação, provocando efeitos colaterais significativos quando alterada

O que é arquitetura de software?

Todo sistema possui uma arquitetura

- Mesmo que não esteja documentada ou seja entendida!

Criadas para atender as necessidades dos *stakeholders*

- Indivíduos, times, organizações ou classes que têm interesse na realização do sistema

Uma boa arquitetura

- Endereça com sucesso as preocupações dos *stakeholders* e, em caso de conflitos, consegue equilibrá-las de maneira aceitável
- Efetivamente e consistentemente comunica os principais aspectos da estrutura do sistema para os *stakeholders* apropriados
  - *"Software architecture documentation speaks for the architect, today, tomorrow, and 20 years from now"*

Sistemas também podem possuir arquiteturas candidatas

- Arranjo particular das estruturas estáticas e dinâmicas que tem potencial para exibir os comportamentos externamente visíveis do sistema e suas propriedades de qualidade
- Trabalho do arquiteto de software selecionar a melhor delas

Arquiteto de software

Responsável pela condução das principais decisões técnicas, que são expressas como a arquitetura de software

Habilidades

Compreender profundamente o domínio e as tecnologias pertinentes

Dominar técnicas de modelagem e metodologias de desenvolvimento

Entender estratégias de negócios da instituição onde atua

Reconhecer estruturas comuns em sistemas já desenvolvidos

Realizar descrição formal da arquitetura de um sistema para analisar as propriedades do sistema

Apresentar arquitetura para outras pessoas

Tarefas

Especificação da arquitetura do software e das bases para o sistema de acordo com requisitos

Análise de *trade-offs* e viabilidade, selecionando alternativa mais adequada ao domínio da aplicação e *stakeholder*

Modelagem da arquitetura, reutilizando e adotando estratégias previamente validadas

Prototipação, simulação e realização de experimentos

Análise de tendências tecnológicas

Atuação como mentor de arquitetos novatos

Estilos arquiteturais

Em geral, sistemas seguem um estilo de organização estrutural

Estilos são “templates” para arquiteturas concretas

- Expressam uma organização estrutural
- Apresentam um conjunto pré-definido de subsistemas e suas responsabilidades
- Inclui regras e diretrizes para organizar o relacionamento entre subsistemas

Mais de um estilo pode ser utilizado em uma mesma aplicação

Estilos arquiteturais

Camadas (*Layers*)

Cliente-Servidor

Tubos e filtros (*Pipes and Filters*)

MVC (*Model-View-Controller*)

Broker

... mas existem outros!

Estilos arquiteturais

Camadas

- Estrutura aplicações que podem ser decompostas em grupos de subtarefas hierarquicamente
- Cada camada oferece serviço à camada acima dela e serve como cliente da camada inferior
- Exemplo: Modelo OSI

Aplicação	Fornecer protocolos comuns para as aplicações
Apresentação	Estruturar a informação
Sessão	Dar suporte para “diálogos” e sincronização
Transporte	Dividir a mensagem em pacotes e garantir a entrega
Rede	Escolher uma rota do remetente ao destinatário
Dados	Detectar e corrigir erros nas seqüências de bits
Física	~~Transmitir bits: velocidade, bit~~-code, conexão, etc.

Estilos arquiteturais

Camadas

Vantagens

Permite projetos baseados em níveis crescentes de abstração

Permite particionar problemas complexos em uma sequência de passos incrementais

Mudanças em uma camada afetam, no máximo, as duas adjacentes

Permite que diferentes implementações da mesma camada possam ser usadas desde que mantenham a mesma interface com as camadas adjacentes

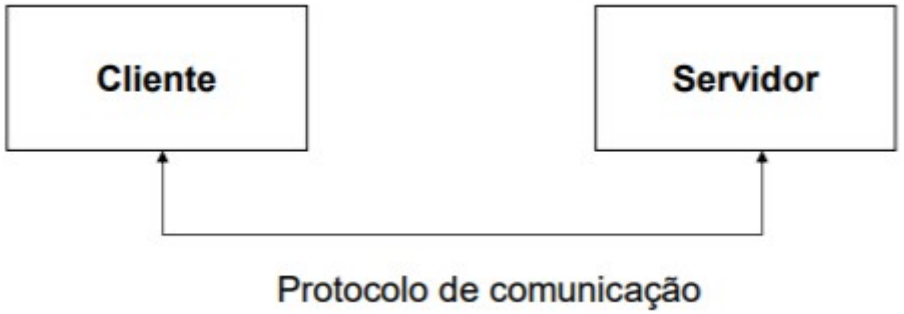
Desvantagens

Nem todos os sistemas são facilmente estruturados em forma de camadas

É difícil encontrar níveis de abstração corretos (muitas vezes, serviços abrangem diversas camadas

Estilos arquiteturais

- Cliente-Servidor
  - Baseado em programas servidores e programas clientes
    - Cliente
      - Estabelece conexão
      - Envia mensagens para servidor
      - Aguarda mensagens de resposta
    - Servidor
      - Aguarda mensagens
      - Executa serviços
      - Retorna resultados



Estilos arquiteturais

Cliente-Servidor

Vantagens

Utilização dos recursos do servidor

Escalabilidade

Aumentando a capacidade computacional do servidor

Desvantagens

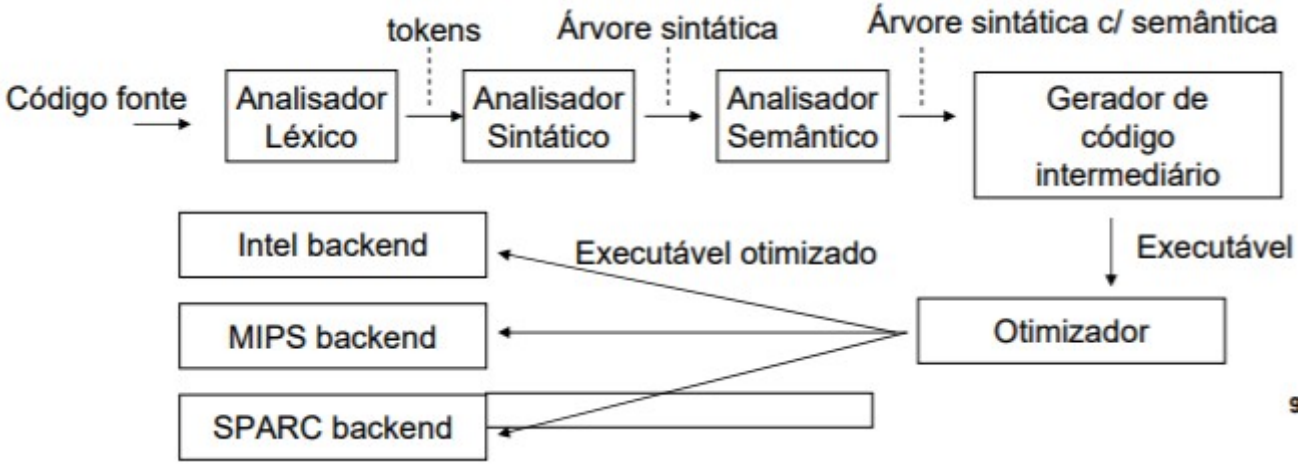
Introduz complexidade

Custos de comunicação

Estilos arquiteturais

Tubos e filtros

- Tipicamente divide a tarefa entre várias etapas de processamento sequencial
- Saída de uma etapa é entrada da etapa seguinte
- Cada etapa de processamento é implementada pelo filtro (filter)
  - Consome e entrega os dados incrementalmente, em vez de todos os dados de uma só vez
- Fluxo de dados entre filtros é implementado pelos tubos
- Exemplo: Compilador



Estilos arquiteturais

Tubos e filtros

Vantagens

Não é preciso criar arquivos intermediários (mas é possível)

Flexibilidade na troca de filtros

Flexibilidade na recombinação

Eficiência no processamento em paralelo

Vários filtros consumindo e produzindo dados em paralelo

Desvantagens

Gerenciamento de erros

Ausência de um estado global compartilhado

Estilos arquiteturais

Model-View-Controller

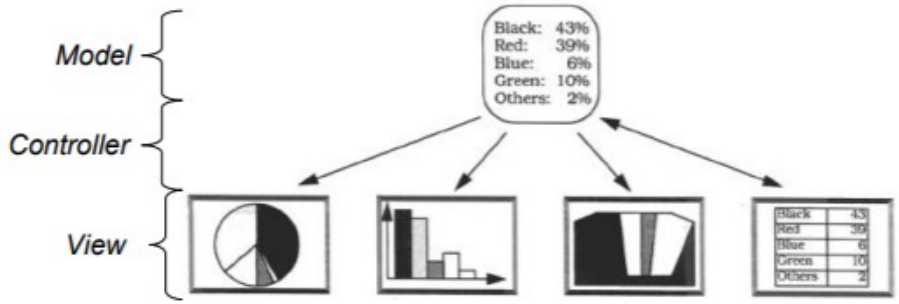
Aplicação dividida em 3 componentes

- Model – contém funcionalidade principal e dados
- View – exibe informação aos usuários
- Controller – gerencia entrada do usuário, deixando o modelo transparente

Interface do usuário = View + Controller

Quando usar?

- Necessidade de várias interfaces com usuário
- Necessidade de várias visões dos dados
- Mudanças nos dados devem ser refletidas na interface



Estilos arquiteturais

Model-View-Controller

Vantagens

Múltiplas "*views*" de um mesmo modelo

"*Views*" sincronizadas

Organização clara de abstrações

Desvantagens

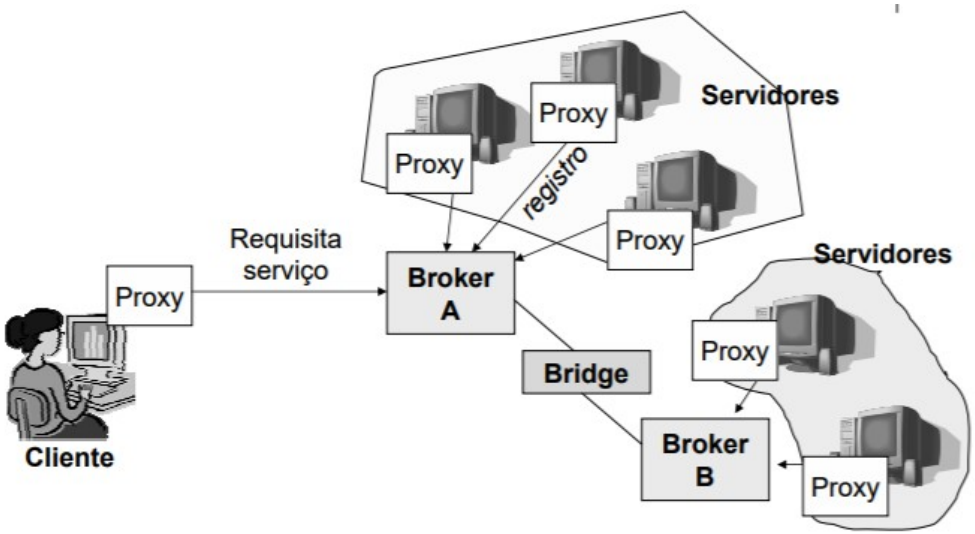
Aumento da complexidade

"*Controllers*" e "*Views*" tendem a ser bastante acoplados

Estilos arquiteturais

Broker

- Estruturar sistemas distribuídos que precisam interagir através de invocação remota de serviços
  - Invocações remotas realizadas via componente *Broker*, que desacopla servidores de clientes e coordenar a comunicação entre componentes
- Exemplos: B2B e Utilização de serviços de busca



Estilos arquiteturais

Broker

Vantagens

Transparência de localização dos serviços

Flexibilidade

Troca de servidores sem alteração nas interfaces não impacta no resto do sistema

Portabilidade

Desvantagens

Sobrecarga de processamento

Debug

Falha na execução de serviço pode ter sido causada tanto pelo cliente quanto pelo servidor
→ Mais variáveis para observar

Estilos arquiteturais

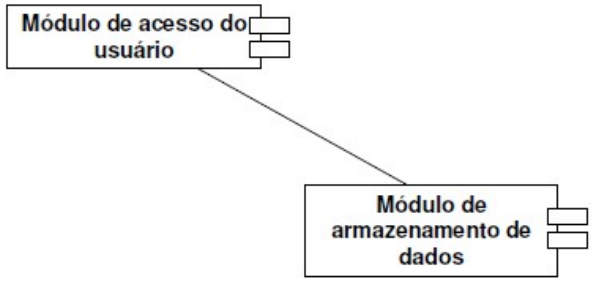
Como selecionar?

- Identificar principais elementos da arquitetura
  - Cada elemento arquitetural tem um estilo arquitetural dominante que reflete as qualidades importantes que devem ser alcançadas no contexto daquele elemento
- Identificar estilo arquitetural dominante
  - O estilo dominante pode ser modificado para alcançar objetivos particulares
  - Se nenhum estilo conhecido parece ser apropriado, o arquiteto deve projetar e documentar um novo estilo
- Considerar responsabilidades adicionais associadas com a escolha do estilo
  - Por exemplo: “Cliente-servidor” demanda a gestão dos protocolos de interação
- Modificar estilo para atingir objetivos adicionais

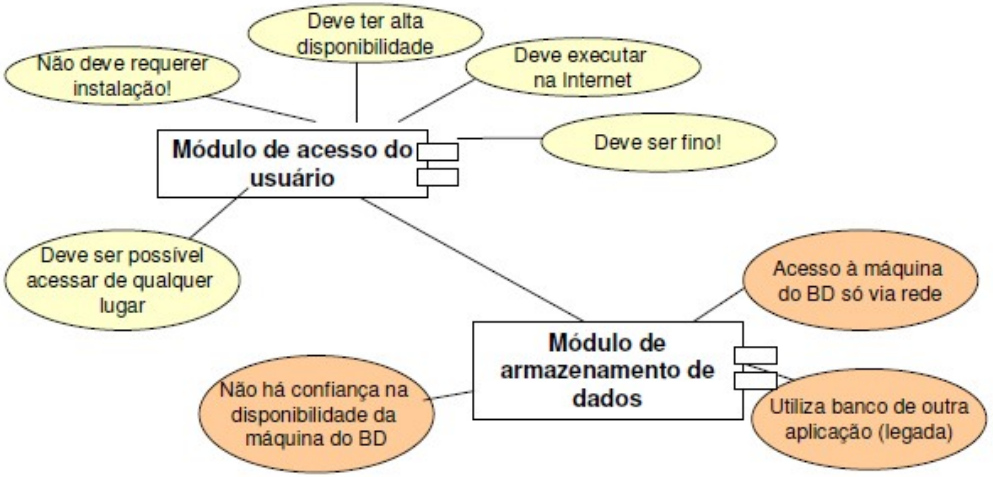
Estilos arquiteturais

Como selecionar? Sistema de matrícula

Identificar principais elementos da arquitetura



Identificar estilo arquitetural dominante



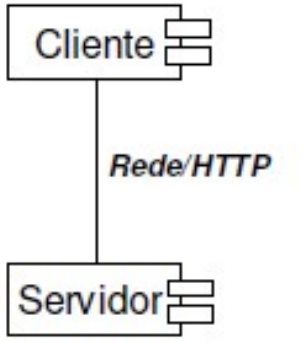
Estilo escolhido: Cliente-servidor

Estilos arquiteturais

Como selecionar? Sistema de matrícula

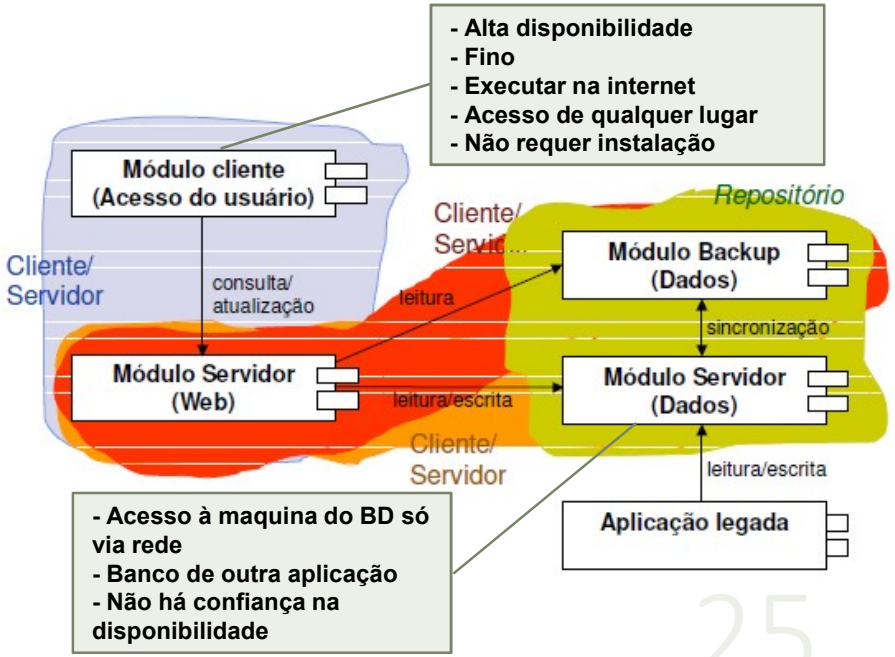
Considerar responsabilidades adicionais

- Estilo dominante escolhido: Cliente-servidor
- Estilo secundário: Repositório
- Responsabilidades: Considerar protocolo de comunicação



Modificar estilo para atingir objetivos adicionais

- Cliente servidor de 2 camadas e repositório com *backup*



Visões arquiteturais

Algumas perguntas precisam ser respondidas durante o projeto da arquitetura:

- Quais são os principais elementos funcionais da arquitetura?
- Como estes elementos irão interagir uns com os outros e com os elementos externos?
- Qual informação será gerenciada, armazenada e apresentada?
- Quais elementos físicos de hardware e software são necessários para suportar estes elementos funcionais e de informação?
- Quais funcionalidades operacionais e capacidades serão disponibilizadas?
- Quais ambientes (teste, suporte, treinamento etc) serão disponibilizados?

Erro comum: Responder todas as perguntas com um único e enorme modelo!

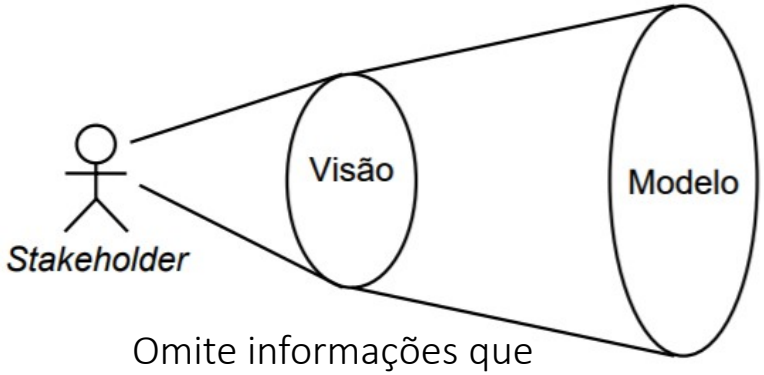
- Difícil representar funcionalidades e propriedades de qualidade de um sistema complexo em um único modelo compreensível para os *stakeholders* → Visões!

Visões arquiteturais

"Representação de um ou mais aspectos estruturais de uma arquitetura que ilustram como essa arquitetura endereça uma ou mais preocupações de seus stakeholders."

IEEE Std 1471 predecessora da IEEE Std 42010

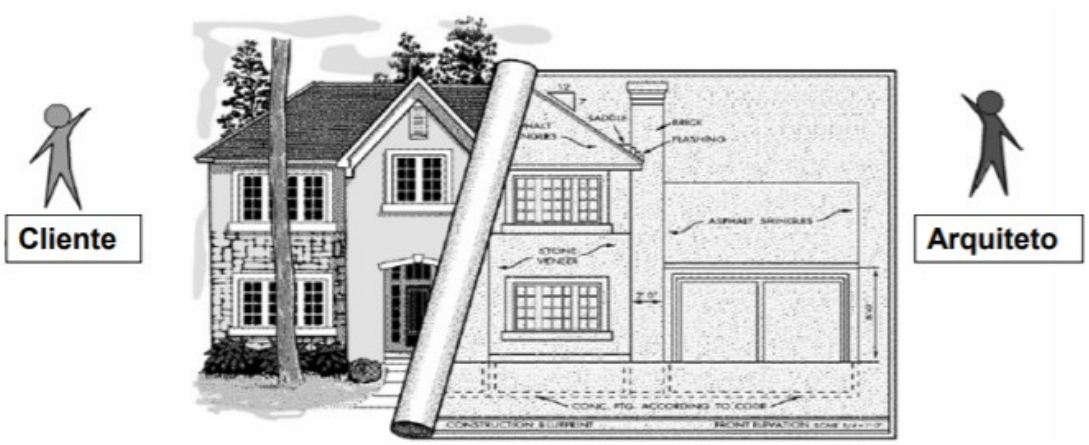
Ou seja, é uma projeção do modelo da arquitetura sob determinada perspectiva, oferecendo diferentes visões para diferentes stakeholders

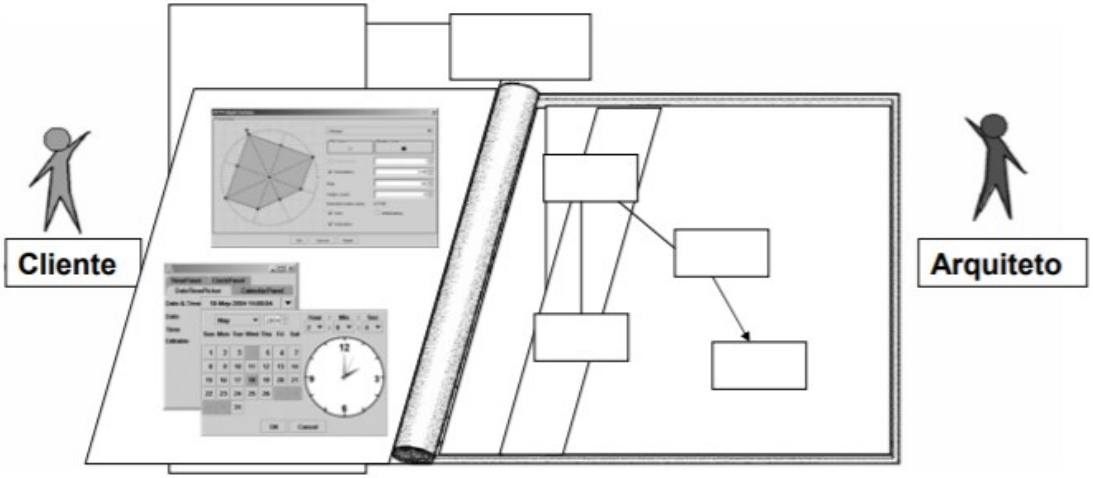


Descrição completa do sistema em um nível específico de abstração

Omite informações que não são relevantes para *stakeholder*

Visões arquiteturais





Visões arquiteturais

Vantagens

Gerenciamento da complexidade

Separação de interesses, facilitando o entendimento

A visão fornecida pelos casos de uso do sistema pode interessar ao cliente

Já a visão de implementação interessa os programadores

Permitem reduzir a quantidade de informação que o arquiteto trata em um dado momento

Desvantagens

Inconsistência

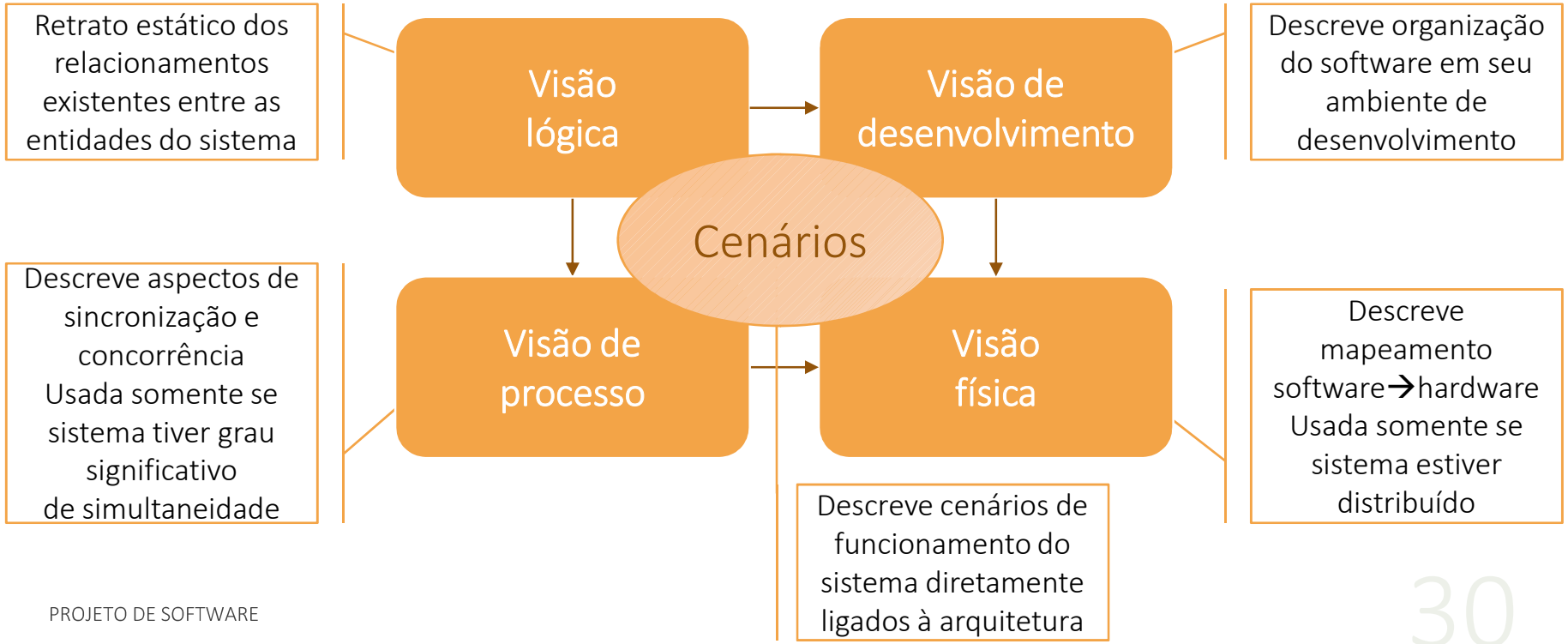
Fragmentação (dificuldade de entender e manter)

Seleção de pontos de vista errados

Visões arquiteturais

Existem diversos conjuntos de visões arquiteturais propostos por diferentes autores

Um dos mais conhecidos: Modelo 4+1 (Kruchten, 1995)



Visões arquiteturais

A descrição de arquitetura não precisa incluir todos os tipos de visões

- Escolher somente aquelas necessárias para responder às preocupações dos *stakeholders*

Duas perguntas que precisam ser feitas para decidir o que incluir:

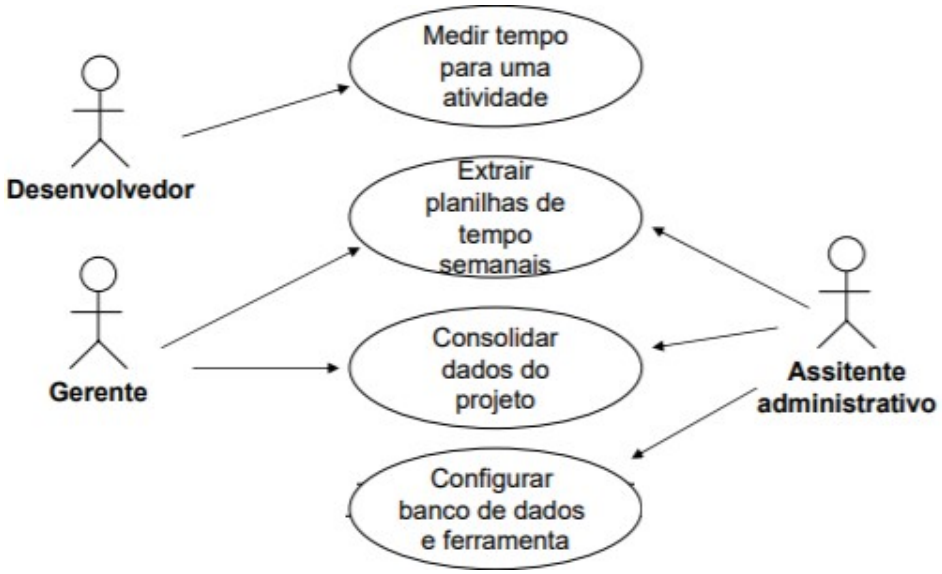
- Os *stakeholders* conseguem utilizar a visão para responder se suas preocupações foram atendidas?
- Os *stakeholders* conseguem utilizar a visão para desempenhar seu papel na construção do sistema?

Descrevendo visões arquiteturais com UML

Cenários

- Não definem a estrutura do sistema
- São necessários para entender como o sistema trabalha e por que a arquitetura é como é

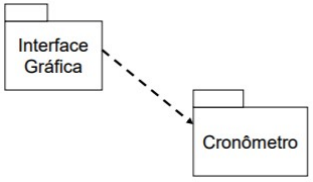
Casos de uso

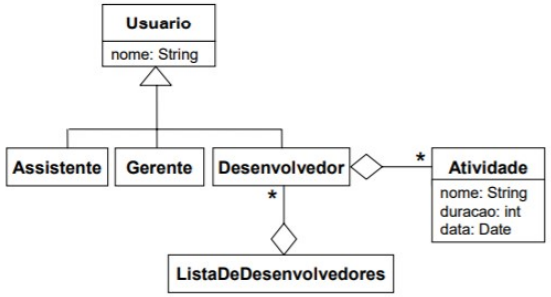


Descrevendo visões arquiteturais com UML

Visão lógica

- Qual a composição lógica do sistema?
- Quais responsabilidades foram atribuídas aos diferentes elementos do sistema?
- Qual agrupamento lógico dos elementos do sistema?



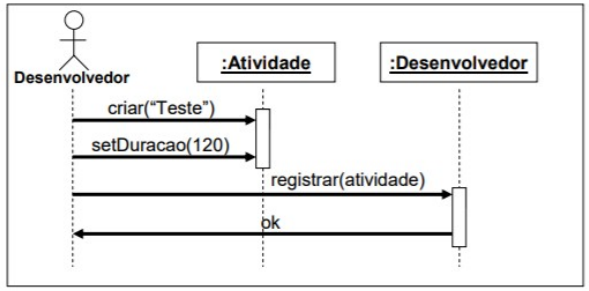


Estrutura

- Diagramas de pacotes
  - Partições maiores
- Diagramas de classes
  - Elementos de design significativos para arquitetura (classes, subsistemas, interfaces) e relacionamentos entre eles

Comportamento

- Diagramas de interação (sequência e colaboração)

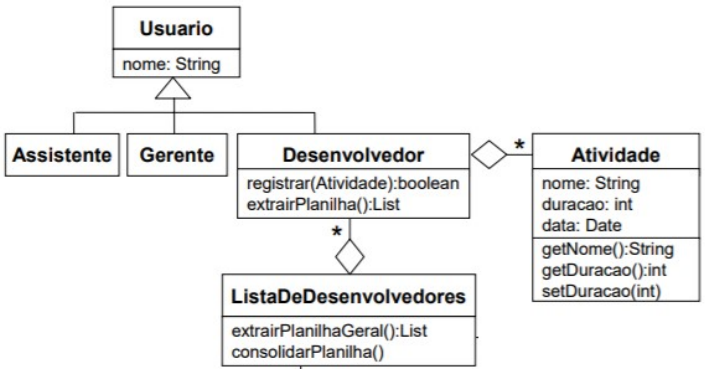


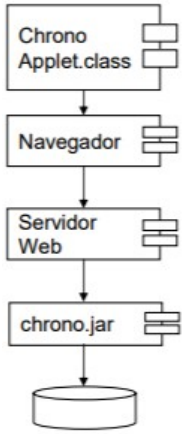
Descrevendo visões arquiteturais com UML

Visão de desenvolvimento

- Quais elementos implementam a instanciação de execução do sistema?
- De onde eles vêm (desenvolvimento interno, compra, etc.)?
- Quais linguagens, técnicas, ferramentas usadas para criá-los?
- Quais competências requeridas para desenvolvê-los/mantê-los?
- Como elementos são organizados em unidades de desenvolvimento e implantação?

- Diagrama de classes (detalhado)
- Diagrama de componentes





Descrevendo visões arquiteturais com UML

Visão de processo

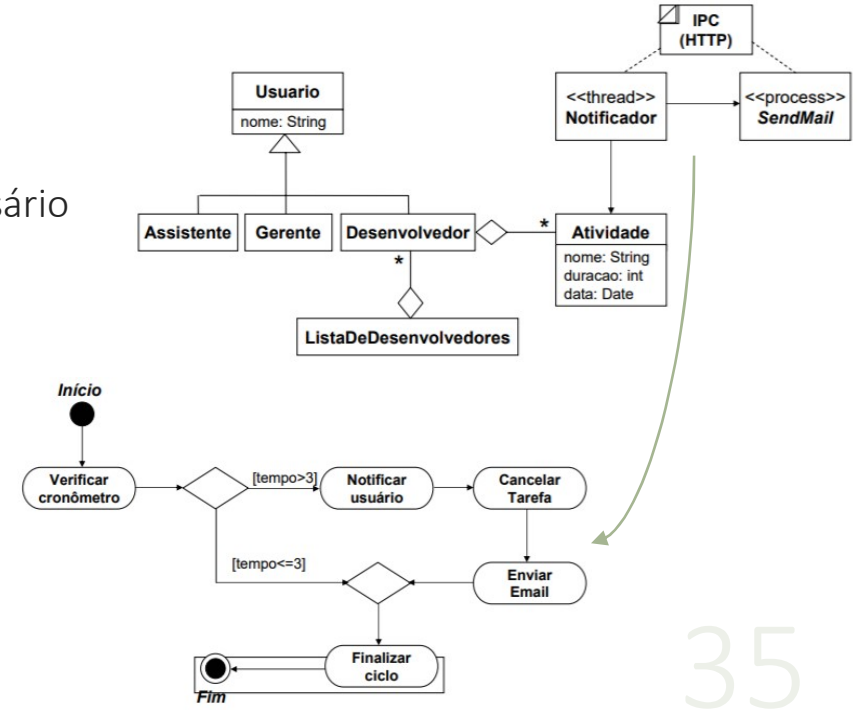
- Quais entidades ativas do sistema?
- Quais threads de controle implementam/executam?
- Quais mecanismos de projeto necessários para comunicação entre objetos?

Estrutura

- Diagramas de Classes
  - Tantos diagramas de classes quanto necessário

Comportamento

- Diagramas de interação
- Diagramas de atividades
- Pelo menos um diagrama por processo

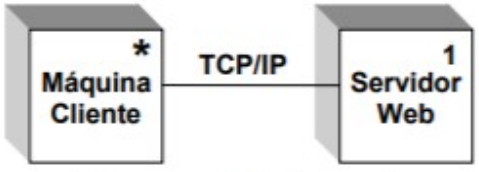


Descrevendo visões arquiteturais com UML

Visão física

- Qual hardware (processadores, elementos de rede, etc.) necessário para executar o sistema?
- Como esses nós serão conectados?
- Quanto isso vai custar?
- Qual será desempenho, confiabilidade e disponibilidade do sistema?
- Como sistema será instalado e mantido?

- Topologia de rede e mapeamento de elementos
  - Diagrama de implantação



Por que arquitetar?

Atua como estrutura a fim de checar o atendimento aos requisitos do sistema

Suporte para estimar custos e gerência da complexidade do sistema

Suporte ao reuso e manutenção

Reduz intervalo entre especificação e implementação

Permite considerar alternativas arquitetônicas em estágios iniciais do desenvolvimento

Abstração base para criar um entendimento mútuo e facilitar comunicação entre participantes

Mas já ouvi que...

Arquitetura = Projeto

- Arquitetura é um aspecto do projeto
  - Foco em elementos importantes estruturalmente e *que têm impacto significativo em desempenho, confiabilidade, custo, adaptabilidade etc.*
- Não diz respeito ao projeto detalhado de componentes individuais

Arquitetura = Infraestrutura

- Infraestrutura é parte integral e importante da arquitetura
- Arquitetura define interoperabilidade entre infraestrutura e componentes da aplicação
- Arquitetura aborda
  - Aspectos dinâmicos
  - Argumentação lógica
  - Adequação ao contexto (de negócio e de desenvolvimento)

Arquitetura é 'plana' e expressa por um único diagrama

- Só é plana em casos muito triviais
- Possui muitas dimensões, que representam múltiplas questões de múltiplos stakeholders
- Uso de um único diagrama para representação de todas essas dimensões leva à sobrecarga semântica → "confusão"
- Requer múltiplas visões

Arquitetura não pode ser medida ou validada

- Avaliadas contra requisitos de qualidade e funcionais, riscos e atributos chave do sistema
- Revisando artefatos da arquitetura
- Testando protótipos da arquitetura

Princípios de Arquitetura de software

Bruna Diirr

brunadiirr@ic.uff.br

Se sobrar tempo...

ArquiteturaOOExemplo.pdf