

# Arquitetura de Software

Donizete C. Bruzarosco

## Software

- **Algumas décadas atrás**
  - Software constituía uma pequena porção dos sistemas computacionais
  - Custos de desenvolvimento e manutenção eram insignificantes quando comparados ao hardware
- **Atualmente**
  - Software constitui uma grande parcela dos sistemas atuais e seus custos cresceram
  - Encontramos software nas mais diversas aplicações:
    - Uso doméstico: processadores de texto, planilhas eletrônicas, etc.
    - Sistemas de grande porte: de geração e distribuição de energia, telecomunicações etc.
    - Empresas têm otimizado suas atividades fazendo uso de sistemas computacionais
  - Software se tornou o elemento chave para o sucesso de qualquer produto
  - O software tornou-se a força prevalecente em termos de inovação tecnológica

## Desenvolvimento de software

- Na construção de sistemas de software principalmente grandes e complexos é essencial tomar uma série de decisões sobre o software e o hardware que serão utilizados
- Para que essas decisões sejam tomadas de forma consciente é preciso conhecer:
  - As funcionalidades do sistema
  - Atributos de qualidade requeridos por ele
- Porém, existem várias questões que precisam ser respondidas
  - Quando cada decisão deve ser tomada?
  - Quem é o responsável pelas decisões?
  - Quais alternativas de soluções devem ser consideradas?
  - Como as alternativas devem ser escolhidas?
- Tais respostas não são óbvias e nem fáceis de se obter

## Desenvolvimento de software Causas de insucesso

- Por pressão do mercado e não utilização de técnicas e métodos de desenvolvimento de software
- Desenvolvedores buscam de imediato a implementação e a infra-estrutura na qual o sistema irá operar, desconsiderando as etapas de análise e projeto de um processo de desenvolvimento de software
- Utilização de novas tecnologias como solução para o desenvolvimento, sem analisar o contexto, o domínio e a adequação ao problema
- Falta de flexibilidade/rapidez para incorporar mudanças inesperadas de requisitos e futuras evoluções, relegando práticas de reutilização
- Preciosismo na especificação de requisitos funcionais, desconsiderando aspectos implícitos da qualidade tais como desempenho, manutenibilidade, segurança, etc.

## Desenvolvimento de software

- Há uma preocupação em resolver esses fatores através de descrições de todos os elementos do sistema, a interação entre estes elementos, os padrões que apoiam tal composição e as restrições, ou seja, a definição de uma arquitetura de software.
- Engenharia de Software (termo cunhado em 1968, numa conferência em Garmisch, Alemanha – Crise de software)
- Muitos métodos têm sido propostos objetivando melhorar o processo de desenvolvimento bem, como minimizar os custos de manutenção
- Na medida que os sistemas tornam-se cada vez maiores e mais complexos a disciplina de arquitetura torna-se fundamental para se obter resultados de baixo custo, atender as restrições de orçamento e cronograma e maior qualidade

## Arquitetura de software

*Com base em discussões realizadas no Software Engineering Institute da Carnegie Mellon University, David Galan e Dewayne Perry definiram arquitetura de software como:*

“A estrutura dos componentes de um programa/sistema, seus inter-relacionamentos, princípios e diretrizes guiando o projeto e evolução ao longo do tempo”.

## Outras definições de arquitetura de software

- Descreve o projeto organizacional de um sistema de software
- Em outras palavras, a arquitetura de um sistema envolve a divisão de funções entre subsistemas ou módulos bem como os mecanismos de interação entre os módulos e a representação da informação compartilhada
- É a estrutura global dos sistema, capturada através da organização do sistema, descrita em elevado nível de abstração, em termos de elementos computacionais pertinentes e das interações entre esses elementos

*OBS.: Neste contexto entende-se componente computacional como sendo uma parte específica do sistema que encapsula um contexto, seus casos de uso, modelos de análise e projeto, especificações de implementação e de teste e forma de interação com outros componentes computacionais. Esse conceito difere de componente de software o qual trata da parte implementável do sistema e que irá se tornar código executável*

## Processo da arquitetura de software

- O termo arquitetura de software é usado para definir duas coisas distintas:
  - o **processo** e
  - o **produto** arquitetural.
- **Processo da arquitetura**  
Compreende as atividades complementares que são necessárias para a construção do software, dentre as quais pode-se destacar:
  - **Elaboração do modelo de negócio para o sistema**  
Para analisar o custo e prazo para o sistema, as restrições de mercado (público alvo) e interfaces com outros sistemas, para se alcançar os objetivos do negócio
  - **Entendimento dos requisitos**  
Utilização de técnicas de levantamento de requisitos p/ se obter o modelo do domínio

## Processo da arquitetura de software

- **Criação ou seleção de uma arquitetura**  
Identificação dos componentes e suas interações, das dependências de construção e da escolha de tecnologias que suportem a implementação
- **Representação da arquitetura e divulgação**  
Os participantes precisam entender a arquitetura e os trabalhos que lhes foi atribuído
- **Implementação do sistema baseado na arquitetura**  
Os desenvolvedores devem se restringir às estruturas e protocolos definidos na arquitetura
- **Análise ou avaliação da arquitetura**  
Deve ser verificada a adequação da arquitetura, registrando impactos, riscos e dificuldades. Tais informações contribuem para a evolução da arquitetura em versões posteriores do sistema

## Produto arquitetural

A partir da aplicação das atividades do processo arquitetural gera-se produtos tais como:

- Modelo do negócio
  - Modelo do domínio de aplicação
  - Modelo dos componentes computacionais e relacionamentos entre eles
  - E a Infra-estrutura tecnológica
- Os quais são considerados modelos de arquitetura

## Importância da arquitetura de software

- **Projeto de software no nível arquitetural envolvem questões tais como:**
  - Organização e estrutura geral de controle
  - Protocolos de comunicação
  - Sincronização
  - Atribuição de funcionalidade a componentes de projeto
  - Escalabilidade e desempenho
  - Seleção de alternativas de projeto

## Importância da arquitetura de software

- Reconhecimento de estruturas comuns de modo que projetistas de software possam compreender as relações existentes entre sistemas e desenvolver sistemas novos com bases nas variações de sistemas antigos
  - O entendimento de arquiteturas de software permite que os engenheiros tomem decisões sobre alternativas de projeto
  - Uma descrição arquitetural do sistema é essencial para analisar e descrever propriedades
  - O conhecimento de notações para descrever arquiteturas possibilita que os engenheiros apresentem novos projetos de sistemas a outros membros de uma equipe de desenvolvimento
- Arquitetura e a comunicação entre os participantes: Cada participante da construção do sistema está preocupado com características específicas que são afetadas pela arquitetura. Portanto sua definição deve ser bem representada, usando uma notação a qual todos os participantes possam entender com facilidade

## Importância da arquitetura de software

- Arquitetura e a antecipação de decisões de projeto
  - As restrições e regras tanto do negócio quanto aquelas que influenciam os aspectos técnicos devem ser atendidas pela arquitetura pois ela constitui-se de um modelo simples e inteligente de como o sistema deve ser estruturado e como os seus componentes trabalham juntos
  - Constitui-se um ponto de referência comum para as demais atividades que são executadas posteriormente a sua definição
  - Preocupa-se com tempo de desenvolvimento, custo e manutenção, definição das restrições de implementação e definição da estrutura organizacional, enfatizando os atributos da qualidade que o sistema requer e medindo através das avaliações a empregabilidade das qualidades necessárias (Bass98)

### •Arquitetura robusta

Após extensas análises, avaliações e revisões da arquitetura, sua representação torna-se robusta o suficiente para guiar o projeto de implementação, os testes e a implantação do sistema

## Benefícios da arquitetura de software

- Atua como uma estrutura a fim de atender aos requisitos de sistema
- Força os projetistas de software a considerar aspectos principais do projeto logo no início
- Ser utilizada como aspecto técnico para o projeto de sistema bem como suporte na estimativa de custos e gerência de processo
- Pode servir como um plano de projeto usado para negociar requisitos de sistema e como um meio de estruturação de discussões com os clientes, desenvolvedores e gerentes
- Servir de base para a análise da consistência e da dependência
- Prover suporte ao reuso
- Ferramenta essencial para gerenciamento de complexidade, pois oculta detalhes e permite um enfoque nas abstrações principais do sistema

## O arquiteto

- Possui uma posição estratégica
- Precisa ter conhecimento profundo:
  - Do domínio onde o sistema a ser desenvolvido será utilizado
  - Das tecnologias relevantes
  - E dos processos de desenvolvimento
- Também deve considerar as implicações que os objetivos organizacionais terão sobre as opções técnicas
- Deverá
  - Construir modelos para o problema a resolver
  - Achar uma solução, explorando abordagens alternativas
  - Gerar a documentação, a qual é considerada imprescindível para a apresentação e discussão com os demais membros da equipe e com o gerente responsável pelo projeto

## Habilidades e tarefas de um arquiteto de software

Habilidades desejadas	Tarefas atribuídas
Compreensão profunda do domínio e das tecnologias pertinentes	Modelagem
Entendimento de aspectos técnicos para desenvolvimento de sistemas bem sucedidos	Análise de compromisso/viabilidade
Técnicas de elicitação, técnicas de modelagem e métodos de desenvolvimento	Prototipação, simulação, realização de experimentos
Entendimento das estratégias de negócios da instituição onde atua	Análise de tendências tecnológicas
Conhecimento de produtos, processos e estratégias de concorrentes	Atuação como mentor de arquitetos novatos

## Outras habilidades inerentes do arquiteto

- Facilidade em trabalhar em vários níveis de abstração
- Capacidade para buscar múltiplas alternativas de projeto
- Ser criativo e fazer investigações

## Técnicas em arquitetura de software

- Aqui são apresentados outros conceitos relacionados à arquitetura de software que, devido à sua importância e contribuição, são utilizados como técnicas que auxiliam a definição de arquitetura (Bass98):
  - Modelo de referência
  - Estilos de arquitetura
  - Arquitetura de referência
  - Arquitetura de linha de produto

## Modelo de referência

- Um modelo de referência consiste na decomposição padronizada do problema em partes conhecidas que cooperam entre si em prol de uma solução. Geralmente, estes problemas são de domínio bastante amadurecido e trazem a experiência de analistas de negócio em conjunto com desenvolvedores [Bass98]. O modelo de referência de um determinado domínio surge durante o processo de amadurecimento da solução em função da necessidade de representações mais abstratas que caracterizam o domínio.
- Um modelo de referência é um framework abstrato para entendimento dos relacionamentos significantes entre as entidades de algum ambiente. Ele habilita o desenvolvimento de arquiteturas específicas usando padrões consistentes ou especificações suportando aquele ambiente. Um modelo de referência consiste de um conjunto mínimo de conceitos unificados, axiomas e relacionamentos com um domínio de um problema particular, e é independente de padrões específicos, tecnologias, implementações, ou outro detalhe concreto.

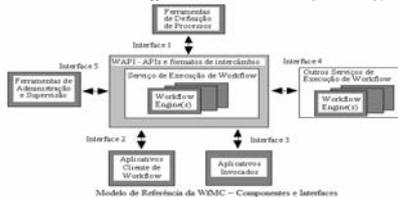
## Modelo de referência

- Como uma ilustração do relacionamento entre um modelo de referência e as arquiteturas que podem derivar de tal modelo, considere o que pode estar envolvido na modelagem que é importante sobre o projeto de uma casa.
- No contexto de um modelo de referência, conhecemos que conceitos tais como
  - áreas de refeição,
  - áreas de higiene e
  - descansosão todos importantes para entender o que compreende uma casa.
- Há relacionamentos entre estes conceitos, e restrições sobre como eles são implementados.

Por exemplo, pode haver separação física entre as áreas de higiene e de refeição.

## Modelo de referência

Workflow Management Coalition (WfMC),



- No modelo de referência da WfMC, apresentado na figura, são definidas cinco interfaces entre componentes, além de uma interface sobre o serviço de execução de workflow, denominada WAPI (Workflow API and Interchange Formats).
- Esta interface consiste em uma série de construções pelas quais os serviços de execução de workflow podem ser acessados. Desta forma, os serviços de workflow podem ser implementados de diferentes formas, contanto que sejam oferecidas interfaces que traduzam os métodos internos de cada produto de workflow para os métodos padronizados pela WfMC.
- Estes padrões de interface estão sendo validados por diversas organizações como a Action Technologies, IBM, FileNet Corporation e a Digital Equipment Corporation.

## Estilos de arquitetura

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, estabelecendo um padrão de utilização (é um padrão de organização de sistemas) [Buschmann96].

**Cada estilo de arquitetura lida com diferentes tipos de atributos da qualidade.**

Para obter a definição de uma arquitetura a partir dos estilos existentes, basta saber quais os atributos mais relevantes para a solução e confrontá-los com os atributos que o estilo atende.

Exemplos de estilos arquiteturais:

- Pipes e filtros
- Camadas
- Objetos
- Quadro negro
- Etc.

## Arquitetura de referência

- Uma arquitetura de referência consiste em componentes de software e os relacionamentos entre eles que implementam funcionalidades relativas às partes definidas no modelo de referência. Cada uma destas partes pode ser implementada em apenas um ou vários componentes de software, ou seja, o mapeamento das funcionalidades do modelo de referência em componentes da arquitetura de referência nem sempre é um para um [Bass98].

**As arquiteturas de referência são aplicáveis a um domínio particular.**

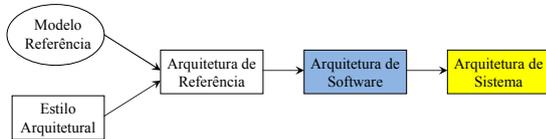
- O papel de uma arquitetura de referência para projeto de uma casa pode ser identificar as soluções abstratas para os problemas de projetar uma casa. Um padrão genérico para projeto de casa, um que enderece as necessidades de seus ocupantes tais como banheiro, cozinha, corredores, e assim por diante é uma boa base para uma arquitetura de referência abstrata.
- O conceito de área de refeição é um conceito no modelo de referência, uma cozinha é a realização de área de refeição no contexto de arquitetura de referência.

## Arquitetura de referência

- Pode haver mais de uma arquitetura de referência que trate de como projetar uma casa, tais como:
- pode haver uma arquitetura de referência que aborde os requisitos para desenvolvimento de soluções para projeto de casas em grandes complexos de apartamentos,
- outro para tratar de casas para uma única família no subúrbio,
- e outra para espaços públicos.
- No contexto de alta densidade de residências, não deve haver uma cozinha separada, mas um espaço de cozinha compartilhada ou ainda uma cozinha comum usada por muitas famílias.

## ARQUITETURA DE SOFTWARE Modelos de Referência, Estilos Arquiteturais, e Arquiteturas de Referência

Modelos de referência, estilos arquiteturais e arquiteturas de referência não são arquiteturas, eles são passos em direção a uma arquitetura.



Uma real – ou concreta – arquitetura pode introduzir elementos adicionais. Ela pode incorporar estilos arquiteturais particulares, arranjos particulares de janelas, materiais de construção a serem usados e assim por diante.

Uma planta de uma casa em particular representa uma instânciação de uma arquitetura como ela é aplicada para a construção de uma moradia real.

## ARQUITETURA DE SOFTWARE Modelos de Referência, Estilos Arquiteturais, e Arquiteturas de Referência

- Os **modelos de referência** agregam solução aos problemas do **ponto de vista de negócio**
- e que **arquiteturas de referência** apresentam a solução do **ponto de vista técnico**, mas baseando-se na **solução de negócio**, ou seja, no modelo de referência determinado para o domínio.
- Um **estilo de arquitetura** também é usado juntamente com o **modelo de referência** para a definição da **arquitetura de referência** que irá apoiar a **definição da arquitetura de software**, como mostra a figura anterior
- Tais técnicas podem ser utilizadas para definir a arquitetura para a construção de um sistema simples bem como a arquitetura para a construção de uma família de sistemas (arquitetura de linha de produtos).

## ARQUITETURA DE SOFTWARE Modelos de Referência, Estilos Arquiteturais, e Arquiteturas de Referência

A maturidade de domínios tais como

- compiladores,
- sistemas gerenciadores de base de dados e
- sistemas operacionais

é vista através da documentação bem padronizada de suas arquiteturas.

Assim sendo, o **trabalho dos arquitetos é simplificado**, pois eles não precisam investir na **definição da arquitetura** e sim na **projeção das propriedades arquiteturais das arquiteturas de referência** que mais se **assemelham a sua necessidade**.

Na maioria das vezes, são adotadas soluções encontradas em livros e manuais que documentam tais arquiteturas.

## Arquitetura de linha de produtos

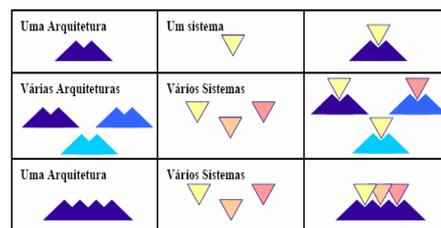
- A criação de uma arquitetura ou a sua manutenção para um sistema que sofre constantes atualizações ou que se diversifica, exige alto investimento de tempo e esforço
- A arquitetura de linha de produtos ou arquitetura de família de sistemas define:
  - Os conceitos
  - Estruturas
  - Componentes e
  - Restrições
 Necessários para obter uma variação de características em vários produtos (ou sistemas) enquanto fornece o máximo de compartilhamento das partes na implementação
- A quantidade de diferenças ou de dependências entre os produtos refletem na complexidade da arquitetura

## Arquitetura de linha de produtos

- Um sistema pode ser representado por vários produtos, estabelecendo-se uma linha de produtos com características e propriedades semelhantes
- Os produtos desta linha podem compartilhar de uma mesma arquitetura construída especificadamente para esta linha em vez de existir uma arquitetura para cada produto
- Construir uma arquitetura para uma linha de produtos significa envolver esforços para maximizar o uso da mesma arquitetura para vários sistemas semelhantes como pode ser observado na figura seguinte

## Arquitetura de linha de produtos

Comparação de arquitetura para construção de um sistema e para construção de vários sistemas



## Arquitetura de linha de produtos

As atividades que envolvem a definição da arquitetura de uma linha de produtos são (jazayeri00):

- Captura de requisitos que impactam na arquitetura  
listar todos os produtos candidatos a participarem da linha, identificar as propriedades genéricas que existem entre produtos candidatos evidenciando as semelhanças, definir o escopo da linha de produtos e a estratégia de produção.
- Analisar a robustez dos produtos candidatos em conformidade com as evoluções futuras: identificar as restrições nos produtos candidatos.
- Projetar as camadas de acordo com níveis de abstração, localizando a arquitetura genérica da linha e a arquitetura de cada produto candidato: projetar também os componentes genéricos.
- Implementar a arquitetura.
- Testar a conformidade da arquitetura, verificando os riscos que afetam o escopo, as propriedades genéricas e a estratégia estabelecida.

## Estilos arquiteturais Introdução

- Durante o período da história do software, diversas arquiteturas foram usadas muitas vezes informalmente
- O objetivo de organizar e expressar o conhecimento do projeto de software de maneira útil é uma necessidade natural do engenheiro (projetista) de software
- Uma forma de codificar tal conhecimento é dispor de um vocabulário do conjunto de conceitos (terminologia, propriedades, restrições), estruturas (componentes, conectores) e padrões de uso existentes
- Ao caracterizar as arquiteturas de software que são utilizadas, identificando seus componentes, mecanismos de interação e propriedades, podemos classificá-las

## Estilos arquiteturais Introdução

- Na década de 90, houve uma crescente preocupação em compreender a organização dos sistemas de software, culminando com o surgimento da área de arquitetura de software
- Identificou-se classes arquiteturais, as quais possuíam aspectos e componentes peculiares a elas bem como formas de combiná-los, caracterizando um estilo arquitetural
- Um estilo arquitetural permite que um profissional determine a classe a qual pertence a organização de um sistema
- Características dos componentes (subsistemas) e conectores do sistema, topologia da arquitetura, restrições semânticas e mecanismos de interação entre os componentes ajudam a identificar o estilo da arquitetura de software do sistema

## Estilos arquiteturais Introdução

- O estilo arquitetural pode ser utilizado como ponto de partida em um projeto, desde que tenhamos em mãos o conjunto de características desejadas para o sistema a ser desenvolvido
- Procura-se identificar qual estilo arquitetural provê suporte à essas características
- Assim, estaria se reutilizando conhecimento e arquitetura de software (já catalogada)
- Cada estilo arquitetural oferece suporte a um conjunto de requisitos não funcionais e atributos de projeto que permitem distinguir uma arquitetura da outra

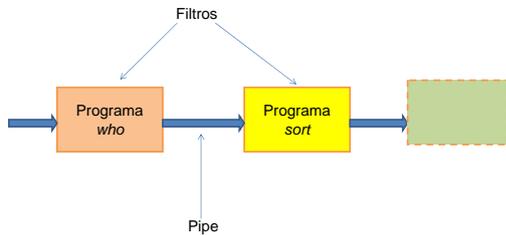
## Estilos arquiteturais Introdução

- Exemplo  
O estilo arquitetural de camadas permite a definição de vários níveis e implica num aumento da flexibilidade do sistema. Entretanto, essa característica decorre em detrimento de outra, isto é, o desempenho do sistema.
- Note que o estilo arquitetural mais apropriado a um sistema dependerá de seus requisitos, envolvendo requisitos funcionais e não funcionais
- Estilos arquiteturais bem definidos com suas características especificadas, permite um menor esforço para entender o projeto de sistema de outra pessoa e, portanto, reduz a quantidade de informações a serem assimiladas num novo projeto
- Deve-se documentar para que se possa consultar e reutilizar conhecimentos e arquiteturas de projetos anteriores

## Estilos arquiteturais PIPES E FILTROS

- Considera a existência de uma rede pela qual flui dados de uma extremidade (origem) à outra (destino)
- O fluxo de dados se dá através de pipes (dutos) e os dados sofrem transformações quando processados nos filtros
- Um pipe (duto) provê uma forma unidirecional de fluxo de dados, uma vez que atua como um condutor para o fluxo de dados entre a fonte até um destino
- O exemplo mais comumente conhecido desse estilo é o utilizado no sistema operacional Unix. A maioria dos usuários desse sistema sabe como canalizar a saída, resultante de um programa (prog1), para a entrada de um outro programa (prog2): exemplo1# prog1 | prog2 - exemplo2# who | sort

## Estilos arquiteturais EXEMPLO DE PIPES E FILTROS

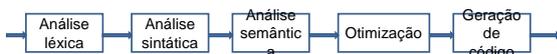


## Estilos arquiteturais PIPES E FILTROS

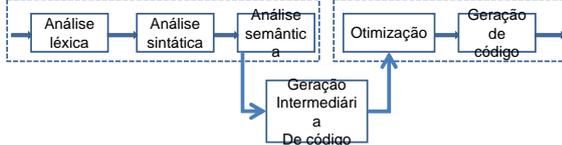
- Uma sequência de pipes e filtros geralmente é chamada de **tubulação de processamento**,
  - **onde os filtros**: incrementam, extraem ou transformam os dados recebidos de alguma fonte de dados (que pode ser outro filtro)
  - **e os pipes**: conectam dois filtros ou alguma fonte de dados e um filtro ou ainda um filtro a algum receptor de dados (extremidade final de uma tubulação de processamento).
- É uma arquitetura adequada para um projeto de sistema que requer **vários estágios de processamento**.  
Cada estágio de processamento seria implementado por um filtro, o qual recebe dados de alguma fonte (podendo ser outro filtro), realiza alguma transformação sobre os dados (extraindo ou acrescentando informações) e produz dados na saída, que são canalizados por meio de pipe ao próximo estágio.

## Estilos arquiteturais EXEMPLO DE PIPES E FILTROS

- **Modelo clássico de compiladores**

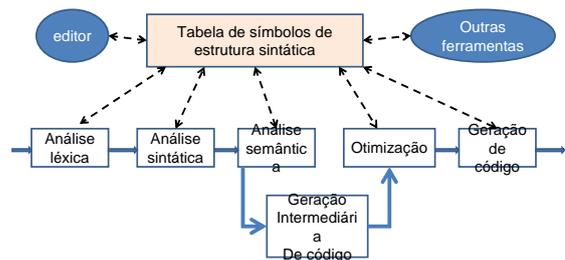


- **Compilador portátil a várias plataformas**



## Estilos arquiteturais EXEMPLO DE PIPES E FILTROS

- **Nova evolução na arquitetura de compiladores**



## Estilos arquiteturais - PIPES E FILTROS Vantagens

- Função do sistema é vista como composição de filtros;
- Problema ou sistema pode ser decomposto de forma hierárquica;
- facilita o entendimento do sistema completo a partir do entendimento de cada filtro isoladamente;
- Manutenibilidade é geralmente mais flexível, possibilitando a reorganização de filtros e pipes
- Facilidade de reuso (código dos filtros), manutenção e extensão, que emprega abordagem caixa preta, onde cada componente tem funcionalidade e interface bem definida, facilitando alterações nos mesmos;
- Desempenho pode ser incrementado através do processamento paralelo de filtros, já que a ativação e uso do componente ocorre com o fluxo de dados, permitindo que componentes com funcionalidades independentes sejam executados de forma concorrente.

## Estilos arquiteturais PIPES E FILTROS

- **Desvantagens**
  - Mudanças frequentes em um componente (filtro) impactar outros componentes. Assim, face a propagação de mudanças de um filtro a outro(s) **a manutenibilidade desse estilo é limitado**.
  - estilo de tubos e filtros coloca ênfase no modo 'batch', tornando difícil seu uso em aplicações interativas
  - Outra questão técnica a ser observada é a possibilidade de haver deadlock com o uso de buffers finitos (para armazenamento temporário de dados)

## Estilos arquiteturais PIPES E FILTROS

Estilo	Pipes e Filtros (Pipes and Filters)	Categoria: From mud to structure
Descrição	Fornece uma estrutura para sistemas que processam uma cadeia de dados. Cada passo é encapsulado em um componente chamado filtro. Os dados passam através dos <i>pipes</i> que ficam entre filtros adjacentes.	
Estrutura	Vantagem	Desvantagem
<ul style="list-style-type: none"> <li>- Os componentes filtro são a unidade de processamento que trata, refina e transforma os dados de entrada.</li> <li>- Os <i>pipes</i> são a conexão entre os filtros.</li> <li>- Os dados de entrada são as entradas do sistema e devem ser do mesmo tipo.</li> <li>- Os dados de saída são o resultado do <i>pipeline</i>.</li> </ul>	<ul style="list-style-type: none"> <li>- Não são necessários arquivos intermediários;</li> <li>- Flexibilidade para troca de filtros;</li> <li>- Flexibilidade para recombinação de filtros.</li> <li>- Reuso de filtros;</li> <li>- Rápida prototipação de <i>pipelines</i>;</li> <li>- Eficiência em processamento paralelo</li> </ul>	<ul style="list-style-type: none"> <li>- Compartilhamento de informações é caro e inflexível;</li> <li>- Ocorre overhead na transformação dos dados</li> <li>- Tratamento de erros é difícil de ser obtido.</li> </ul>

## Estilos arquiteturais - Camadas

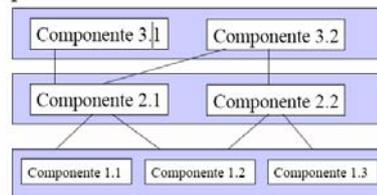
- Estrutura um sistema num conjunto de camadas, onde cada uma delas agrupa um conjunto de tarefas num determinado nível de abstração
- Uma camada no nível N oferece um conjunto de serviços à camada no nível superior (N+1). A camada N faz uso dos serviços disponíveis na camada inferior (N-1)
- Um exemplo típico de uma arquitetura de camadas é o modelo de referência OSI (Reference Model for Open Systems Interconnection) da ISO

## Estilos arquiteturais – Camadas Arquitetura de camadas do modelo OSI da



## Estilos arquiteturais – Camadas

- Cada camada individual pode ser uma entidade complexa consistindo de diferentes componentes



## Estilos arquiteturais – Camadas

- Não há uma especificação de qual deveria ser a granularidade de componentes. Entretanto, componentes complexos, geralmente exigirão decomposição adicional
- Faz a decomposição do sistema em um conjunto de camadas, onde cada camada acrescenta um nível de abstração sobre a camada inferior
- O número de camadas depende da funcionalidade a ser oferecida pelo sistema
- Necessidade de definir critérios de abstração para agrupar subtarefas para comporem uma camada. Exs: tipos de componentes da aplicação (específico) e de interface com o sistema operacional

## Estilos arquiteturais – Camadas Variações sobre o estilo de camadas simples

- Camada N utiliza os serviços oferecidos pela camada N-1 para realizar suas funções  
Um exemplo de variação desse estilo seria permitir que a camada N tivesse acesso também as camadas N-2, N-3 ou a qualquer outra abaixo dela.
- Essa variação na arquitetura de camadas pode comprometer a manutenibilidade de um sistema.
- Um maior grau de dependência entre as camadas implica que mais de uma camada necessitará ser modificada para atender à mudança de requisito

## Estilos arquiteturais – Camadas

- A definição de vários níveis de abstração (camadas), proporciona uma maior flexibilidade e suporte a portabilidade do sistema
- Desde que sua interface permaneça inalterada, uma camada poderá ser substituída por outra equivalente
- Quando a interface de uma camada é alterada ou novos recursos são adicionados, somente a camada adjacente é afetada
- Ainda possui um custo associado:  
O desempenho da arquitetura de camadas fica comprometido face à necessidade de uma solicitação externa ao sistema precisar passar por várias camadas a fim de ser tratada.

## Estilos arquiteturais – Camadas

- Sistemas em camadas mantêm as dependências de máquina em camadas mais internas
  - isso torna mais fácil fornecer implementações de várias plataformas de um sistema de aplicação
  - Somente as camadas mais internas dependentes de máquina, precisam ser reimplantadas para levar em conta os recursos de um sistema operacional ou banco de dados diferente
- Implementar um sistema como um bloco monolítico não constitui uma solução apropriada

## Estilos arquiteturais – Camadas

- Apóia o desenvolvimento incremental de sistemas  
A medida que uma camada é desenvolvida, alguns serviços fornecidos por essa camada podem ser disponibilizados aos usuários
- A solução (desafio) é buscar uma arquitetura com menor número de camadas que atenda aos requisitos de desempenho, mas que ao mesmo tempo considere a manutenibilidade
- Não é só a funcionalidade a ser provida pelo sistema que determina o número de camadas, mas também os requisitos não-funcionais desejados (desempenho, manutenibilidade, etc.)
- **Determinar o número adequado de camadas de um sistema é uma tarefa muito difícil**

## Estilos arquiteturais – Camadas

- Também pode-se chegar à conclusão que o estilo arquitetural de camadas não constitui uma solução adequada e, assim, deve-se identificar um outro estilo que satisfaça aos requisitos funcionais e não funcionais.
- Considere, por exemplo, a arquitetura OSI, a qual possui sete camadas
- Contudo, devido ao surgimento de diferentes tecnologias tanto a fim de atender a usuários que necessitam de redes de alta velocidade numa área específica quanto de conectar equipamentos separados a milhares de quilômetros de distância, houve nova ênfase na arquitetura de redes.
- Com isso surgiu a arquitetura Internet TCP/IP

## Estilos arquiteturais – Camadas

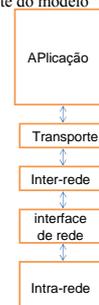
- A arquitetura de camadas permite o reuso de camadas
- Por exemplo, TCP (Transmission Control Protocol) pode ser usado em diferentes aplicações, tais como telnet e ftp
- Um outro exemplo desse estilo compreende os sistemas Web de múltiplas camadas que separa cliente, servidores de aplicação, servidores Web e outros clientes Web.

## Estilos arquiteturais – Camadas

### Arquitetura Internet TCP/IP

- É estruturada em quatro camadas construídas sobre uma quinta camada (intra-rede) que não faz parte do modelo

Com a arquitetura Internet foi possível a interligação de redes de computadores com tecnologias distintas



A solução foi agrupar os níveis físico, de enlace e de rede da arquitetura OSI na camada intra-rede havendo uma interface entre essa e a camada inter-rede

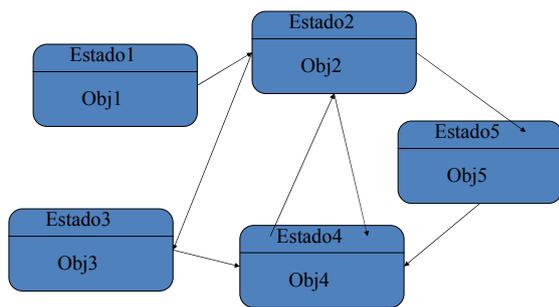
## Estilos arquiteturais – Camadas

Estilo	Camada (Layer)	Categoria: From mud to structure
Descrição	Auxilia na aplicação de estruturas que podem ser decompostas em grupos de subtarefas pertencentes a um nível particular de abstração.	
Estrutura	Vantagem	Desvantagem
<ul style="list-style-type: none"> <li>- Cada camada pode ser composta por vários componentes que interagem entre si ou com os componentes da camada seguinte.</li> <li>- As requisições são movidas do nível mais alto para o nível mais baixo. As respostas para as requisições ou notificações de eventos trafegam no sentido oposto.</li> </ul>	<ul style="list-style-type: none"> <li>- Reuso de camadas;</li> <li>- Suporte a padronização;</li> <li>- Dependências mantidas localmente.</li> </ul>	<ul style="list-style-type: none"> <li>- Mudança de comportamento em cascata (que ocorre quando uma mudança afeta mais de uma camada);</li> <li>- Baixa eficiência;</li> <li>- Impacto no desempenho;</li> <li>- Dificuldade em estabelecer a granularidade correta das camadas.</li> </ul>

## Estilos arquiteturais – objetos

- O paradigma orientado a objetos acrescenta uma nova abstração ao projeto de software
- Combina numa única entidade (objeto) tanto os dados quanto as funções que atuam sobre esses dados
- O estilo arquitetural de objetos tem como base o uso de um tipo de dados abstrato – o objeto, o qual possui várias propriedades importantes tais como:
  - **Objetos são entidades independentes que podem sofrer modificação uma vez que toda informação pertinente é mantida no próprio objeto.**
  - É possível fazer o mapeamento entre as entidades reais e os objetos que atuam no controle de um sistema, resultando numa melhor compreensão dos sistema
  - **Os objetos fazem uso de um mecanismo de troca de mensagens para se comunicar em vez de utilizar variáveis compartilhadas**
  - Objetos podem ser reutilizados devido a sua independência
  - **Os objetos podem estar distribuídos e executar sequencialmente ou em paralelo, a depender das decisões tomadas no início do projeto**
  - **Possui como base a ocultação da informação**

## Estilos arquiteturais – objetos



## Estilos arquiteturais – objetos

- O estilo arquitetural de objetos vê um sistema de software como um conjunto de objetos comunicantes com estado associado a eles.
  - Quando o objeto *obj1* necessita se comunicar com um objeto *obj2*, ele precisa conhecer a identidade do objeto ao qual enviará uma mensagem.
  - Numa arquitetura de objetos, teríamos um conjunto de objetos com estados próprios (escondidos) e as operações associadas àqueles estados
  - Os objetos (comunicantes) podem requisitar ou oferecer serviços a outros objetos
  - É comum utilizar a arquitetura orientada a objetos em **sistemas de informação** como sistemas de consulta e empréstimos online de bibliotecas de instituições de ensino que dispõem de componentes de cadastro de usuários e componentes de autenticação de usuários.
- Note que componentes similares existem em outros sistemas de informações, tais como sites de conteúdos (jornais e revistas) que exigem cadastro e autenticação de qualquer usuário antes de disponibilizar o conteúdo.*

## Estilos arquiteturais – objetos

Embora os conceitos do paradigma orientado a objetos possa ser utilizado para tratar alguns aspectos do projeto arquitetural, existem algumas diferenças entre as características e benefícios do projeto orientado a objetos e o projeto da arquitetura de software :

- Projeto de arquitetura de software
  - Visa decompor o sistema em componentes e identificar as interações existentes entre esses componentes
  - Essa decomposição em componentes fornece uma visão global do sistema, permitindo que o engenheiro/arquiteto de software analise e raciocine sobre as propriedades e restrições do sistema
- Dado que os estilos arquiteturais podem descrever famílias de projetos parece intuitivo vislumbrar o paradigma de projeto orientado a objetos como um estilo arquitetural no qual todos os componentes são objetos, e todas as conexões são associações ou agregações

## Estilos arquiteturais – objetos

- Existem questões que são relevantes , as quais são tratadas na abordagem orientada a objetos, mas que fogem ao escopo do projeto arquitetural, tais como: **formas para modelar requisitos**, bem como o **projeto de estrutura de dados e algoritmos** ; os quais não precisariam ser tratados ou especificados numa descrição arquitetural
- Se a manutenibilidade for um requisito não-funcional preponderante , é fundamental considerar que qualquer **modificação de requisitos** deveria afetar o **menor número possível de objetos**
- Se o arquiteto de software tiver de levar em conta **desempenho do sistema** , então os cenários de uso mais frequentes deveriam ocorrer num **menor número de objetos** a fim de minimizar a computação de contextos que viria a comprometer o desempenho do sistema.

### Estilos arquiteturais Invocação implícita

- A idéia por trás de invocação implícita é que ao invés de invocar um procedimento diretamente (objeto), um componente pode anunciar (ou difundir) um ou mais eventos.
- Outros componentes no sistema podem registrar um interesse em um evento associando um procedimento a ele.
- Quando o evento é anunciado, o sistema invoca todos os procedimentos que tenham sido registrados para o evento.
- Um evento sendo anunciado implicitamente causa a invocação de procedimentos em outros módulos.

### Estilos arquiteturais Invocação implícita

- Um exemplo de sistema empregando mensagens são listas de notícias e fórum que possuem componentes de registro de novos usuários acoplados ao componente de autenticação.
- Perceba que esse tipo de sistema apenas permite que o usuário tenha acesso ao conteúdo se este for devidamente autenticado e registrado.

### Estilos arquiteturais Invocação implícita -Vantagens

- Um benefício importante de invocação implícita é que ela fornece forte suporte para reuso. Qualquer componente pode ser introduzido no sistema simplesmente por registrá-lo a eventos daquele sistema.
- Outro benefício é que invocação implícita facilita evolução do sistema. Componentes podem ser substituídos por outros componentes sem afetar as interfaces de outros componentes no sistema.

### Estilos arquiteturais Invocação implícita - Desvantagem

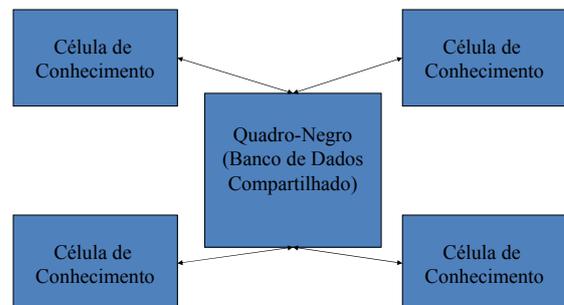
- Uma desvantagem de invocação implícita é que os componentes abrem mão do controle sobre a computação desempenhada pelo sistema
  - Quando um componente anuncia um evento, ele não pode assumir que outros componentes responderão a esse evento. Mesmo que ele conheça quais outros componentes estão interessados nos eventos que ele anunciou, ele não pode interferir na ordem em que eles são invocados
  - Os componentes não podem fazer qualquer suposição sobre o tipo de computação a ser realizada ou ordem de processamento.

### Estilos Arquiteturais – Quadro-Negro

- Originou-se na IA (usado, por ex., para compartilhamento de conhecimento)
- Considera a existência de um repositório central de dados, circundado por um conjunto de componentes (células de conhecimento)
- Tem como base um modelo de solução de problema que fornece uma estrutura conceitual para organizar o conhecimento do domínio, bem como uma estratégia para aplicar esse conhecimento
- Vários agentes colaboram para resolver um problema.
- O quadro negro é uma entidade ativa capaz de notificar aos agentes periféricos quando há algum trabalho novo que poderia ser realizado por um agente específico
- Isoladamente, nenhum agente sabe resolver o problema sozinho mas trabalhando em conjunto, cada um avança um aspecto distinto e o problema é resolvido.

### Estilos Arquiteturais – Quadro-Negro

É constituído de 3 elementos:  
células de conhecimento, estrutura de dados do quadro e controle



## Estilos Arquiteturais – Quadro-Negro

- É adequado em aplicações onde diversos tipos de conhecimentos devem ser considerados a fim de dar suporte à interpretação de um conjunto de dados iniciais.
- Tipicamente era utilizado em casos onde não havia soluções gerais para um problema.

Nesse caso, um ou mais componentes (célula de conhecimento) interagem com o banco de dados compartilhado (quadro negro) buscando encontrar uma solução parcial ou total

- Esta arquitetura mais simples envolve um único repositório central de dados (quadro negro). Entretanto, problemas mais complexos podem exigir múltiplos repositórios organizados num sistema distribuído

## Estilos Arquiteturais – Quadro-Negro

- Os componentes interagem como quadro-negro buscando encontrar uma solução total ou parcial.
- Facilidade de adicionar ou remover tipos de dados, os componentes podem ser adicionados ou removidos sem acarretar modificações em outros componentes.
- Dá suporte à manutenibilidade.
- Desempenho comprometido pela necessidade de um componente (célula de conhecimento) “passear” pelo repositório de dados (quadro-negro) em busca de uma solução.

## Estilos Arquiteturais – Quadro-Negro

Estilo	Blackboard	Categoria: From mud to structure
Descrição	É útil para problemas que não possuem estratégias de soluções determinísticas conhecidas e são baseados em soluções aproximadas ou parciais. São caracterizados por problemas que, quando decompostos em subproblemas, abrangem muitos campos de conhecimento. A solução para problemas parciais necessita de diferentes paradigmas e representações.	
Estrutura	Vantagem	Desvantagem
<ul style="list-style-type: none"> <li>- O sistema é dividido em: um componente <i>blackboard</i>, uma base de conhecimentos e um componente de controle.</li> <li>- O <i>blackboard</i> é uma central de armazenamento dos dados – o vocabulário.</li> <li>- O componente de controle monitora as mudanças no <i>blackboard</i> e decide qual ação deve ser tomada.</li> <li>- A base de conhecimentos consiste em subsistemas separados e independentes, cada qual resolvendo aspectos específicos do problema.</li> </ul>	<ul style="list-style-type: none"> <li>- Ajuda a resolver problemas de experimentação;</li> <li>- Suporte a mudanças e manutenção;</li> <li>- Reuso de conhecimentos;</li> <li>- Suporte a tolerância a falhas e robustez.</li> </ul>	<ul style="list-style-type: none"> <li>- Dificuldades para testar por não ter algoritmos determinísticos;</li> <li>- Nenhuma boa solução é garantida;</li> <li>- Dificuldade em estabelecer uma boa estratégia de controle;</li> <li>- Baixa eficiência e alto esforço de desenvolvimento;</li> <li>- Não suporta paralelismo.</li> </ul>

## Estilos Arquiteturais Sistemas orientados a eventos

- São regidos por eventos gerados externamente
- Evento
  - Pode ser um sinal que pode assumir uma gama de valores
  - ou uma entrada de comando baseados em um menu
- A diferença entre um evento e uma entrada simples é que a ocorrência do evento está fora de controle do processo que manipula esse evento
- Modelos de controle orientados a evento
  - Broadcast: um evento é transmitido a todos os subsistemas. Qualquer subsistema programado para manipular esse evento pode responder a ele
  - Orientado a interrupções: são usados exclusivamente em sistemas de tempo real, nos quais interrupções externas são detectadas por um tratador de interrupções. Estas são, então passadas para algum outro componente para processamento

## Estilos Arquiteturais Sistemas orientados a eventos – Modelo broadcast

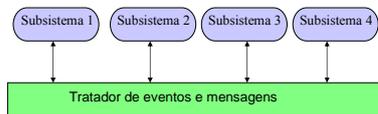
- Os subsistemas registram um interesse em eventos específicos
- Quando esses eventos ocorrem, o controle é transferido para o subsistema que pode tratar o evento
- Os subsistemas decidem de quais eventos necessitam e o tratador de eventos e mensagens assegura que esses eventos sejam enviados a eles
- Todos os eventos podem ser transmitidos a todos os subsistemas, mas isso impõe um grande overhead de processamento
- Mais frequentemente, o tratador de eventos e mensagens mantém um registro dos subsistemas e dos eventos interessados neles

## Estilos Arquiteturais Sistemas orientados a eventos – Modelo broadcast

- Subsistema gera evento
- Tratador de eventos detecta os eventos, consulta o registrador de eventos e passa o evento aos subsistemas que declararam interesse
- Vantagens
  - Evolução é relativamente simples
  - Um novo subsistema para tratar classes específicas de eventos pode ser integrado por meio do registro de seus eventos no tratador de eventos
  - Qualquer subsistema pode ativar qualquer outro subsistema sem saber seu nome ou sua localização
  - Os subsistemas podem ser implementados em máquinas distribuídas
- Desvantagens
  - Subsistemas não sabem se ou quando os eventos serão manipulados
  - É possível que subsistemas diferentes se registrem para os mesmos eventos, podendo causar conflitos quando os resultados de manipulação de eventos forem disponibilizados

## Estilos Arquiteturais Sistemas orientados a eventos – Modelo broadcast

### Modelo de controle baseado em broadcast seletivo



## Estilos Arquiteturais – Outros estilos

- Os estilos arquiteturais discutidos anteriormente constituem os mais conhecidos pela sua utilização nos mais variados sistemas.
- Outros, entretanto, também são encontrados em diversas aplicações tais como Repositório (passivo), arquiteturas de aplicações distribuídas.

## Arquiteturas de aplicações distribuídas

- Há considerável discordância na literatura sobre o que constitui um sistema distribuído
- Dentre as diversas definições encontradas, há um ponto de concordância : a **presença de múltiplos processadores**
- Os dois estilos arquiteturais mais comuns
  - multiprocessadores: vários processadores autônomos compartilhando uma memória primária comum – apropriado para executar diversas subtarefas de um mesmo programa
  - Multicomputadores: arquitetura similar a de multiprocessadores, exceto que os processadores não compartilham memória. A comunicação se dá através de passagem de mensagens numa rede de comunicação (programas concorrentes)

## Arquiteturas de aplicações distribuídas

- Uma aplicação distribuída possui quatro tipos de processos:
  - Filtros: são transformadores de dados
  - Cliente : pode ser visto como um processo – na interação entre cliente e servidor, o cliente é o componente que inicia alguma atividade
  - Servidor: é um processo reativo uma vez que reage às solicitações feitas pelos clientes
  - Par (peer): é um dentre um conjunto de processos idênticos que interage a fim de oferecer algum serviço ou realizar alguma computação (ex: em programação paralela, no qual vários pares interagem a fim de resolver um problema)

## Estilos da Arquitetura de aplicações distribuídas

- **Processos comunicantes:** utilizados quando as metas prioritárias do sistema a ser desenvolvido são escalabilidade e facilidade de modificações (ex. de aplicação: quando se tem um conjunto de trabalhadores (componentes computacionais) replicados que compartilham um repositório de tarefas – esse estilo é utilizado em programação paralela.
- **Cliente-servidor:** as tarefas são divididas entre produtores e consumidores de dados.
  - **Servidor:** é um processo que fica num estado de espera, aguardando solicitação de serviço de um ou mais clientes (pode trabalhar de forma síncrona ou assíncrona)
  - **Cliente:** processo pode estar no mesmo sistema ou em algum outro, é conectado ao servidor via rede. Os clientes podem ser vistos como processos que atuam de forma independente, isto é, a execução de um processo não interfere em outro(s).
    - Facilidade de remover ou adicionar clientes
    - Facilidade de modificar a funcionalidade de um cliente (visto que outros clientes não serão afetados)

## Estilos da Arquitetura de aplicações distribuídas

- **Chamada de Procedimento Remoto**
  - É um mecanismo de transferência de controle de um processo (chamador) para outro (chamado), sendo depois o controle retornado ao processo que emitiu a chamada
  - A chamada de procedimento remoto permite a comunicação bidirecional. Um dos objetivos é aumentar o desempenho do sistema ao distribuir a computação a ser realizada entre vários processadores.

## Computação interorganizacional distribuída Arquitetura ponto a ponto

- Sistemas ponto a ponto (p2p) são sistemas descentralizados em que as computações podem ser realizadas por qualquer nó da rede e, em princípio pelo menos, nenhuma distinção é feita entre clientes e servidores
- O sistema global é projetado para beneficiar-se da capacidade computacional e armazenamento disponíveis em uma rede computadores potencialmente grande
- Os padrões e protocolos que possibilitam as comunicações através dos nós estão embutidos na aplicação, e cada nó deve realizar uma cópia dessa aplicação
- Tipos de arquiteturas
  - descentralizada:
  - semicentralizada

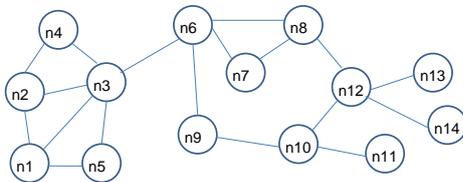
## Computação interorganizacional distribuída Arquitetura ponto a ponto

### Exemplos

- Sistemas de compartilhamento de arquivos baseados em protocolos Gnutella e Kazaa são usados para compartilhar arquivos em PCs de usuários, e
- sistemas de mensagens instantâneas, como ICQ e Jabber, permitem comunicação direta entre os usuários sem um servidor intermediário
- Existem indicações de que essa tecnologia está cada vez mais sendo usada pelas empresas para aproveitar a potência de suas redes de PC
- A Intel e a Boeing implementaram sistemas p2p para aplicações que requerem computação intensa

## Computação interorganizacional distribuída Arquitetura ponto a ponto descentralizada

- não são simplesmente elementos funcionais, mas também chaves de comunicação que podem guiar os sinais de dados e de controle de um nó para outro

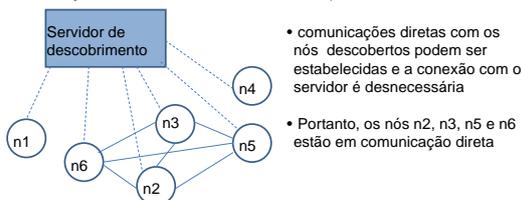


## Computação interorganizacional distribuída Arquitetura ponto a ponto descentralizada

- Suponha que a figura representa um sistema descentralizado de gerenciamento de documentos
- Esse sistema é usado por um consórcio de pesquisadores que compartilham documentos e cada membro do consórcio mantém seu próprio repositório de documentos
- Quando um documento é recuperado por um nó, ele também o disponibiliza para os outros nós
- Alguém que necessite de um documento emite um comando de pesquisa que é enviado aos nós naquela "localidade"
- Esses nós verificam se eles têm o documento e, caso tenham, retornam o documento ao solicitante.
- Portanto, se n1 emite uma busca por um documento armazenado em n10, essa pesquisa é guiada através dos nós n3, n6, e n9 e n10
- **Vantagens:**
  - Altamente redundante e, assim, tolerante a defeitos e tolerante quanto aos nós que se desconectam da rede
- **Desvantagens:**
  - Overheads: comunicações replicadas entre pares; mesma busca pode ser processada por muitos nós diferentes

## Computação interorganizacional distribuída Arquitetura ponto a ponto semicentralizada

- O papel de um servidor é auxiliar a estabelecer contato entre os pares na rede ou coordenar os resultados de uma computação
- A figura representa um sistema de mensagens instantânea
- Os nós da rede se comunicam com o servidor (linhas tracejadas) para encontrar quais outros nós estão disponíveis
- Comunicações diretas com os nós descobertos podem ser estabelecidas e



- comunicações diretas com os nós descobertos podem ser estabelecidas e a conexão com o servidor é desnecessária
- Portanto, os nós n2, n3, n5 e n6 estão em comunicação direta

## Computação interorganizacional distribuída

- Em um sistema p2p no qual uma computação intensiva de processador é distribuída entre um grande número de nós, é normal que alguns nós – cujo papel é distribuir trabalho para outros nós e comparar e verificar os resultados da computação – destaquem-se
- Embora haja *overhead* em sistemas ponto a ponto ela é considerada uma abordagem eficiente para computação interorganizacional
- Problemas com essa abordagem (não resolvidas):
  - Proteção
  - Confiança
- São mais adequados a sistemas de informações não críticos ou nos quais já existam relacionamentos de trabalho entre as organizações

## Arquitetura de aplicação distribuída Arquitetura de sistema orientada a serviços

- O desenvolvimento da web promoveu o acesso de computadores clientes aos servidores remotos fora de suas próprias organizações
- Organizações convertiam suas informações em HTML tornando-as acessíveis por esses computadores
- Contudo, o acesso era somente por meio de navegador web, e o acesso direto aos repositórios de informações por outros programas não era prático
- Isso impossibilitou as conexões convenientes entre os servidores nas quais, por exemplo, um programa consulta uma série de catálogos
- Para contornar esse problema, a noção de um web service foi proposta

## Arquitetura de aplicação distribuída Arquitetura de sistema orientada a serviços

- Usando esse tipo de serviço, as organizações que desejam tornar acessíveis suas informações a outros programas podem fazer isso com a definição e publicação de uma interface de Web service
- Essa interface define os dados disponíveis e como eles podem ser acessados
- **Genericamente um web service é uma representação padronizada de alguns recursos computacionais e de informações que podem ser usadas por outros programas**
- Por exemplo: você pode definir um serviço de declaração de impostos no qual os usuários podem preencher seus formulários de impostos e estes serem verificados automaticamente e enviados às autoridades fiscais
- **Um web service é uma instância de uma noção mais geral de um serviço**

*Uma ação ou serviço oferecido de um grupo para um outro. Embora o processo possa estar ligado a um produto físico, o desempenho é essencialmente intangível e não resulta normalmente em propriedade de*

## Arquitetura de aplicação distribuída Arquitetura de sistema orientada a serviços

- A essência de um serviço, portanto, é que o fornecimento dos serviços é independente da aplicação que usa o serviço
- Os provedores de serviços podem desenvolver serviços especializados e oferecê-los a uma gama de usuários de serviços de organizações diferentes
- As aplicações podem ser construídas pela ligação de serviços de vários provedores que usam uma linguagem padrão de programação ou uma linguagem de harmonização de serviços especializados como BPEL4WS
- Existem vários modelos de serviços, do modelo JINI (Kumaran, 2001) a Web services (Stal, 2002), e Grid services (Foster, et al., 2002).
- Conceitualmente, todos os modelos funcionam conforme a figura seguinte

## Arquitetura de aplicação distribuída Arquitetura de sistema orientada a serviços



- Um provedor de serviços oferece um serviço pela definição de sua interface e pela implementação da funcionalidade do serviço
- Um solicitante do serviço vincula esse serviço a sua aplicação. Isso significa que a aplicação do solicitante inclui códigos para chamar o serviço e processa os resultados da chamada
- Para assegurar que o serviço possa ser acessado por usuários de serviços externos, o provedor de serviços faz uma entrada em um registro de serviço que inclui informações sobre o serviço e o que ele faz

## Arquitetura de aplicação distribuída Arquitetura de sistema orientada a serviços

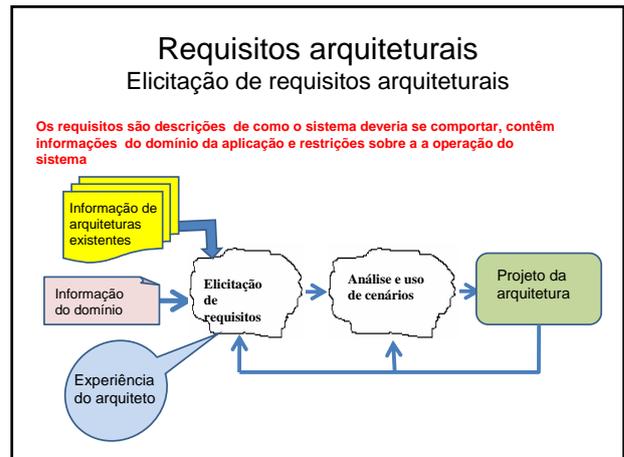
- Os serviços podem ser oferecidos por qualquer provedor de serviços dentro ou fora da organização
- O provedor de serviços torna públicas as informações sobre o serviço de maneira que usuários autorizados podem usá-los
- A criação conveniente de novos serviços é possível
- Os usuários de serviços podem pagar pelos serviços de acordo com o uso no local de fornecimento
- As aplicações podem se tornar menores
- As aplicações podem ser reativas e adaptar sua operação de acordo com o ambiente por meio da vinculação de serviços diferentes à medida que o ambiente muda

## Arquitetura de aplicação distribuída Arquitetura de sistema orientada a serviços

- Web services é uma base para a construção de aplicações distribuídas não firmemente acopladas
  - Há ainda uma experiência prática limitada quanto a arquitetura orientada a serviços e, assim, ainda não se sabe quais são as implicações práticas dessa abordagem.
- Existem três padrões fundamentais que possibilitam comunicações entre Web services:
- **SOAP** (Simple Object Access Protocol): define uma organização para troca estruturada de dados entre Web services
  - **WSDL** (Web Services Description Language): define como as interfaces dos Web services podem ser representadas
  - **UDDI** (Universal Description Discovery and Integration): é um padrão de descobrimento que define como as informações de descrição do serviço, usadas pelos solicitantes do serviço para descobrir serviços, pode ser organizada.



Estilo	Broker	Categoria: Sistemas Distribuídos
<b>Descrição</b> É utilizado em sistemas cuja estrutura é distribuída com desacoplamento de componentes que interagem através de invocação remota de serviços. O componente <i>broker</i> é responsável por coordenar a comunicação.		
<b>Estrutura</b>		
<ul style="list-style-type: none"> <li>- O <i>servidor</i> possui objetos que expõem sua funcionalidade através de interfaces com operações e atributos.</li> <li>- Os <i>clientes</i> são aplicações que acessam os serviços de pelo menos um <i>servidor</i>.</li> <li>- O <i>broker</i> é mensageiro responsável pela transmissão de requisições do cliente para o <i>servidor</i> e transmissão de respostas e exceções do <i>servidor</i> para o cliente.</li> <li>- Os <i>proxies</i> são componentes que escondem detalhes de implementação quando dois <i>brokers</i> interopem.</li> <li>- O <i>proxy</i> do lado do cliente é a camada entre cliente e <i>broker</i> que esconde detalhes de implementação do cliente tais como a transparência na qual um objeto remoto aparece para o cliente como local, mecanismos de comunicação usados para transferência entre cliente e <i>broker</i>, criação e remoção de blocos de memória e <i>marshaling</i> de parâmetros e resultados.</li> <li>- O <i>proxy</i> do lado do <i>servidor</i> é análogo ao <i>proxy</i> do lado do cliente e é responsável por receber requisições, desempacotar mensagens, <i>unmarshaling</i> parâmetros e chamar serviços apropriados.</li> </ul>		
<b>Vantagem</b>		
<ul style="list-style-type: none"> <li>- Transparência para o cliente na localização do <i>servidor</i>.</li> <li>- Facilidade de troca e evolução de componentes uma vez que suas interfaces não mudam.</li> <li>- Portabilidade em função do <i>broker</i>.</li> <li>- Interoperabilidade entre diferentes <i>brokers</i>.</li> <li>- Reuso dos serviços já existentes quando da construção de novas aplicações cliente.</li> </ul>		
<b>Desvantagem</b>		
<ul style="list-style-type: none"> <li>- Eficiência restrita uma vez que os <i>servidores</i> são localizados dinamicamente.</li> <li>- Baixa tolerância a falhas.</li> <li>- Teste e <i>debug</i> são dificultados por envolver muitos componentes.</li> </ul>		



## Requisitos arquiteturais

- O processo de desenvolvimento de software baseado na arquitetura, considera a arquitetura de software como fator orientador do processo
- A complexidade de um sistema de software é determinada tanto por seus requisitos funcionais (o que ele faz) quanto por requisitos não funcionais (de qualidade)

Para obter a definição de uma arquitetura a partir dos estilos existentes, basta saber quais os atributos mais relevantes para a solução e confrontá-los com os atributos que o estilo atende.

**Cada estilo de arquitetura lida com diferentes tipos de atributos da qualidade.**

## Requisitos arquiteturais

Influência dos requisitos não funcionais (atributos da qualidade como requisitos críticos)

A arquitetura do sistema afeta, por exemplo, o desempenho, facilidade de distribuição e de manutenção. O estilo e a estrutura específicos escolhidos para uma aplicação podem depender dos requisitos não funcionais do sistema:

- **Desempenho:** a arquitetura deve ser projetada para localizar operações críticas dentro de alguns subsistemas, com a menor comunicação possível. Isso pode significar usar componentes de alta granularidade
- **Proteção:** uma estrutura de camadas para a arquitetura pode ser usada, com os itens mais críticos protegidos por camadas mais internas e com um alto nível de validação de proteção aplicado a essas camadas
- **Segurança:** as operações relacionadas à segurança devem estar localizadas em um único subsistema ou em um pequeno número de subsistemas. Isso reduz os custos e os problemas de validação de segurança e torna possível fornecer esse serviço a sistemas de proteção relacionados
- **Disponibilidade:** a arquitetura deve ser projetada para incluir componentes redundantes e a possibilidade de substituir e atualizar componentes sem parar o sistema
- **Facilidade de manutenção:** a arquitetura deve ser projetada usando componentes de baixa granularidade que possam ser prontamente mudados

## Requisitos arquiteturais

### Atributos de projeto

- Princípios que norteiam o processo de desenvolvimento de software
- **Separação de interesses**
  - A separação de interesses permite lidar com diferentes aspectos de um problema, de modo que possamos não concentrar em cada um deles isoladamente. Isso permite lidar com a complexidade inerente.
  - Tem sido aplicada sempre que um sistema é decomposto em vários módulos de modo que a arquitetura do sistema contenha mais de um componente.
  - Alguns dos aspectos de interesse são a funcionalidade oferecida, confiabilidade esperada e desempenho

## Requisitos arquiteturais - Atributos de projeto

- **Separação de interesses**
  - Aqui o interesse recai em fazer a separação em termos da qualidade ou requisitos não funcionais desejados. Por exemplo, num sistema de software poderia-se querer lidar com desempenho e grau de correção separadamente.
    - Nesse caso o projetista poderia inicialmente decidir fazer o projeto e estruturar o software de modo que o grau de correção pudesse ser assegurado a priori.
    - Depois o software seria reestruturado levando em conta o desempenho desejado

## Requisitos arquiteturais - Atributos de projeto

- **Abstração**
  - Permite entender questões complexas de projeto e resolver problemas
  - Concentrar em um nível mais genérico sem se ater a detalhes irrelevantes
  - Possibilita trabalhar num problema, usando conceitos e termos que sejam familiares no domínio do problema antes de transformá-los em alguma estrutura de mais baixo nível
  - Pode haver diferentes abstrações da mesma realidade, cada uma delas, fornecendo uma visão da realidade e atendendo a algum propósito específico
  - O engenheiro de software constrói modelos que são analisados
  - Geralmente emprega-se diagramas que podem ter algum formalismo associado, tal como a OCL (Object Constraint Language)

## Requisitos arquiteturais - Atributos de projeto

- **Modularidade**
  - Permite a separação de interesses
  - atributo da qualidade de suma importância para o projeto de software
  - Minimiza o impacto de mudanças de uma decisão sobre outra(s)
  - Gerencia a complexidade do projeto
  - Dá produtividade ao desenvolvimento
  - A efetividade da organização do sistema dependerá do(s) critério(s) usado(s) para decompor o sistema em componentes (coesão/acoplamento)

## Requisitos arquiteturais - Atributos de projeto

- Existem três metas que a modularidade tenta obter:
  - Capacidade de decompor um sistema complexo e/ou de grande porte; Dividir um problema em subproblemas e replicar tal procedimento de decomposição a cada subproblema recursivamente
  - Capacidade de compor um sistema a partir de um conjunto de módulos  
Projetistas compõem o sistema a partir de componentes elementares que incrementalmente vão sendo agregados até a obtenção de um sistema completo
  - Compreender a modularização do sistema  
Entender cada parte de um sistema, separadamente, facilita modificações que se façam necessárias ao sistema

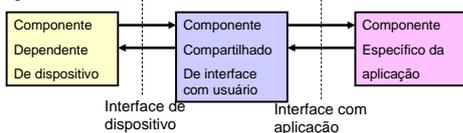
## Requisitos arquiteturais - Atributos de projeto

- **Compartilhamento de recursos**  
Os critérios usados para modularizar um sistema são alta coesão e baixo acoplamento
  - O acoplamento é usado como medida de interdependência entre dois componentes, exemplos:
    - Componente A chamar a rotina de um componente B
    - Componente A ter acesso a uma variável declarada no componente B
  - Se dois componentes possuem elevado nível de acoplamento, tornar-se-á difícil de analisá-los, compreendê-los, modificá-los ou até mesmo reutilizá-los
  - No entanto, pode-se obter um acoplamento relativamente baixo entre componentes de um sistema quando compartilhando recursos. Geralmente há um gerenciador de recursos que é encarregado de designar quem terá acesso aos recursos compartilhados.

## Requisitos arquiteturais - Atributos de projeto

### Compartilhamento de recursos

- Para exemplificar considere o modelo arquitetural de software de interface com o usuário da figura



- **Componente específico da aplicação:** código específico ao programa de uma aplicação. Incorpora o núcleo funcional da aplicação e pode incluir o código da interface com o usuário que é específica da aplicação
- **Componente compartilhado de interface com o usuário:** código que provê suporte à interface com o usuário de múltiplos programas de aplicação. Se o sistema pode acomodar diferentes dispositivos de E/S, conterá apenas a parte do código que é associado aos tipos de dispositivos
- **Componente dependente de dispositivos:** compreende o código que é pertinente a uma classe específica de dispositivos de E/S, não sendo específico da aplicação

## Projeto da arquitetura do software

•O produto do processo de projeto de arquitetura é um documento de projeto de arquitetura. Ele pode incluir várias representações gráficas do sistema junto com um texto descritivo associado

•Os modelos gráficos do sistema apresentam perspectivas diferentes da arquitetura

•Os modelos de arquitetura podem incluir (Sommerville, 2007):

- **Modelo estático de estrutura:** que mostra os subsistemas ou componentes desenvolvidos como unidades separadas
- **Modelo dinâmico de processo:** que mostra como o sistema está organizado em processos em tempo de execução
- **Modelo de interface:** que define os serviços oferecidos por cada subsistema por meio de suas interfaces públicas
- **Modelos de relacionamentos:** que mostram os relacionamentos, tal como o fluxo de dados entre os subsistemas
- **Modelo de distribuição:** que mostra como os subsistemas podem ser distribuídos pelos computadores.

## Linguagens de descrição de arquitetura

- Alguns pesquisadores propuseram o uso de linguagens de descrição de arquiteturas (ADL – Architectural Description Language) para descrever arquiteturas de sistemas
  - **Exemplos de ADLs:** ACME (CMU/USC), Rapide (Stanford), Wright (CMU), Unicon (CMU) Aesop (CMU), MetaH (Honeywell), C2 SADL (UCI), SADL (SRI)
- Os elementos básicos das ADLs são componentes e conectores, e eles incluem regras e guias para arquiteturas bem formadas
- **Pontos positivos**
  - Representam um meio formal para a representação da arquitetura
  - São projetadas para serem legíveis tanto pelas máquinas quanto por pessoas
  - Suportam descrever um sistema em um nível mais alto do que anteriormente era possível
  - Permitem análises das arquiteturas – completude, consistência, ambigüidade, e desempenho
  - Podem suportar geração automática de software

## Linguagens de descrição de arquitetura

- **Pontos negativos**
  - Porém ADLs podem ser somente compreendidas por especialistas em linguagens e são inacessíveis para especialistas em domínios e aplicações (Sommerville, 2007), tornando-as difíceis de analisar com base em uma perspectiva prática e, com isso, levando-as a serem utilizadas em um pequeno número de aplicações
  - Representações atualmente em uso são relativamente difíceis de serem analisadas sintaticamente e não são suportadas por ferramentas comerciais
  - A maioria dos trabalhos sobre ADL são mantidos com metas acadêmicas em detrimento dos objetivos práticos
- Notações como a **UML** permanecerão como as mais comumente usadas para descrição de arquiteturas (Sommerville, 2007).

## Arquitetura em Sistemas de Informação

- Uma classe de sistemas com grande quantidade de aplicações computacionais que envolve armazenamento, recuperação e processamento de dados é chamada de sistemas de informação.
- Seu principal objetivo é o gerenciamento da informação
- Um dos componentes mais importantes dessa classe de sistemas é o banco de dados
- Dentre os atributos da qualidade que caracterizam um SI, pode-se destacar
  - Desempenho
  - Segurança
  - Integridade
  - Disponibilidade
  - Usabilidade (para aplicações interativas)

## Arquitetura em Sistemas de Informação

- Algumas das principais preocupações no desenvolvimento de SI
  - Possibilitar o compartilhamento entre muitos usuários
  - Flexibilidade – para permitir mudanças constantes
  - Fator-chave: manutenção – permite lidar com mudanças frequentes
- **Arquitetura de software – um aspecto determinante de sucesso**

## Alternativas de arquiteturas para Sistemas de Informação

### Arquitetura cliente-servidor

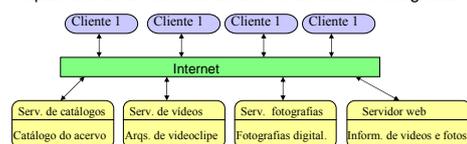
Utilizadas em sistemas de grande porte com acesso a dados bem como de transações

- Diversas variações da tecnologia cliente-servidor têm evoluído, porém compartilhando o conceito de possuir a funcionalidade de processamento compartilhado entre uma máquina cliente e uma máquina servidora.
- Principais componentes
  - **Cliente:** recupera dados do servidor (solicita serviços de servidores)
  - **Servidor:** aceita conexões e comandos oriundos de clientes e retorna respostas e dados aos clientes (oferece serviços, por ex: servidor de impressão, de arquivos, etc.)
  - **Rede:** permite acesso pelos clientes aos servidores, exceto quando ambos estiverem na mesma máquina

## Alternativas de arquiteturas para Sistemas de Informação

### Arquitetura cliente-servidor

- Arquitetura de um sistema de acervo de filmes e fotografias



- Tradicionalmente a tecnologia cliente-servidor pode ser implementada adotando-se as seguintes arquiteturas:
  - Arquitetura cliente-servidor de duas camadas
  - Arquitetura cliente-servidor de múltiplas camadas



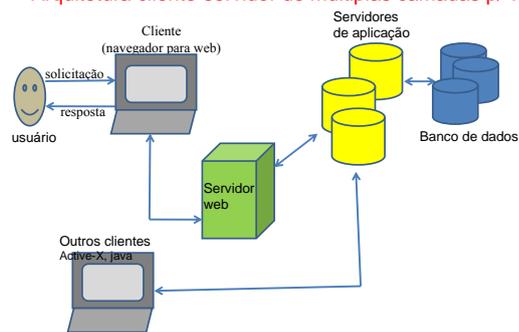
### Alternativas de arquiteturas para Sistemas de Informação Arquitetura cliente-servidor de múltiplas camadas

- Geralmente é desenvolvida com base em arquiteturas proprietárias que têm como desvantagens:
  - Elevado custo de aquisição/licenciamento e manutenção
  - Processo de desenvolvimento, geralmente complexo e dispendioso
  - Requer pessoal altamente especializado para manutenção do sistema
  - Dificuldade para integração num ambiente heterogêneo, envolvendo múltiplos vendedores

### Alternativas de arquiteturas para Sistemas de Informação Arquitetura cliente-servidor de múltiplas camadas p/ WEB

- Redesenhou a forma na qual os negócios têm sido realizados em virtude de uma infra-estrutura de distribuição de informações
- Informações residem em um servidor central acessível em toda a web a partir de qualquer computador
  - navegador
  - conexão com a internet
- Vantagens:
  - Baixo custo de manutenção;
  - Acesso universal através de navegadores para clientes
  - Acesso de qualquer local
  - Etc.

### Alternativas de arquiteturas para Sistemas de Informação Arquitetura cliente-servidor de múltiplas camadas p/ WEB



### Alternativas de arquiteturas para Sistemas de Informação Arquitetura cliente-servidor de múltiplas camadas p/ WEB Exemplo

Um sistema de operações bancárias pela internet é um exemplo de arquitetura cliente-servidor de três camadas:

- o banco de dados de clientes (geralmente hospedado em um computador mainframe) fornece serviços de gerenciamento de dados;
- um servidor web/aplicação fornece os serviços de aplicação, como recursos para transferir dinheiro, gerar extratos, pagar contas, etc.;
- e o próprio usuário com um navegador de internet é o cliente.

### Alternativas de arquiteturas para Sistemas de Informação Arquitetura em Sistemas de tempo real

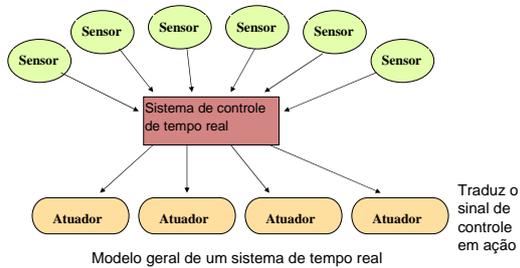
- **Principal característica**  
É que eles devem responder aos estímulos recebidos dentro de um estrito intervalo de tempo
- Um sistema de tempo real é um sistema de software cujo funcionamento correto depende dos resultados produzidos pelo sistema e do tempo em que esses resultados são produzidos
- **Sistema de tempo real leve:** é aquele cuja operação será degradada caso os resultados não sejam produzidos de acordo com os requisitos de *timing* especificados
- **Sistema de tempo real rígido:** é aquele cuja operação será incorreta se os resultados não forem produzidos de acordo com a especificação de *timing*.

### Alternativas de arquiteturas para Sistemas de Informação Arquitetura em Sistemas de tempo real

- **Estímulos:**
    - **Estímulos periódicos:** ocorrem em intervalos de tempo previsíveis  
Por ex.: um sistema pode examinar um sensor a cada 50 milissegundos e reagir de acordo com o valor desse sensor (o estímulo)
    - **Estímulos aperiódicos:** ocorrem irregularmente e são geralmente sinalizados por meio do mecanismo de interrupção do computador.  
Por exemplo: um estímulo seria a interrupção que indique que uma transferência de E/S foi concluída e os dados estão disponíveis em um *buffer*
  - **Arquitetura:**  
Deve ser organizada de modo que, tão logo um estímulo seja recebido, o controle seja transferido para o tratador correto.
- A generalidade desse modelo de estímulo-resposta conduz a um modelo de arquitetura genérica e abstrata, no qual existem três tipos de processos:
- **Processo de gerenciamento de sensor:** para cada tipo de sensor
  - **Processos computacionais:** calculam as respostas exigidas para os estímulos recebidos pelo sistema
  - **Processos de controles de atuadores:** gerenciam a operação de atuadores

## Alternativas de arquiteturas para Sistemas de Informação Arquitetura em Sistemas de tempo real

- Modelo sensor-sistema-atuador de um sistema embutido de tempo real



## Alternativas de arquiteturas para Sistemas de Informação Arquitetura em Sistemas de tempo real

- As **linguagens de programação** para desenvolvimento de sistemas de tempo real precisam incluir recursos para acessar o hardware do sistema e deve ser possível prever o **timing de operações específicas** nessas linguagens
- Linguagens de baixo nível: (permitem gerar código eficiente)
  - Assembler
  - C
- Exemplo:
  - Sistema de comutação eletrônica o qual possui requisitos precisos de tempo real e confiabilidade
  - Software embarcado: deve reagir a eventos gerados pelo hardware e emitir sinais de controle em resposta a esses eventos
  - Outros: Sistemas de navegação para veículos; Controle de tráfego aéreo; Sistemas de monitoramento de vida; Controle de processos industriais; Sistemas distribuídos de multimídia.